UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería de
Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

Desarrollo de un sistema de detección de tráfico anómalo en redes SCADA basadas en el protocolo IEC 60870-5-104

David Abreu Cañamares
Tutor: Jorge Enrique López de Vergara Méndez

JULIO 2018

Desarrollo de un sistema de detección de tráfico anómalo en redes SCADA basadas en el protocolo IEC 60870-5-104

AUTOR: David Abreu Cañamares
TUTOR: Jorge Enrique López de Vergara Méndez

High Performance Computing and Networking Research Group
Dpto. de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio de 2018

Resumen

La familia de estándares IEC 60870-5 define los protocolos utilizados habitualmente para telecontrol en redes SCADA. Dichas redes se encargan de la supervisión, control y adquisición de datos de relevancia en procesos industriales controlados, tales como aquellos que forman parte de la producción energética o tratamiento de recursos para el beneficio humano. Durante el último lustro, el número de amenazas a la seguridad en la red ha aumentado exponencialmente, hasta convertirse en uno de los principales problemas para empresas u organizaciones, tanto públicas como privadas. Las redes SCADA no son una excepción, siendo de especial importancia el despliegue de medidas de seguridad que permitan el correcto funcionamiento de los procesos que controlan. La tendencia actual a la interconexión de la mayoría de los sistemas desplegados, o la gestión de incidencias de manera remota, provoca la aparición de múltiples vulnerabilidades, fallos de seguridad y filtraciones en los datos o recursos de cualquier ente. En concreto, en este TFG se ha estudiado la seguridad en el entorno de un sistema industrial de supervisión, control y adquisición de datos sobre el protocolo IEC 60870-5-104, que se implementa sobre redes TCP/IP, y la realización de un IDS que consigue detectar dichas intrusiones en la red. Este tipo de redes SCADA se caracterizan por tener una baja tasa de intercambio de paquetes junto a una muy alta regularidad en el intervalo de las peticiones al servidor, lo cual permite diseñar soluciones de detección de amenazas en tiempo real una vez lograda la reproducción de la caracterización de la red. El sistema IDS implementado se encarga de la captura de los paquetes, del análisis de estos paquetes mediante el estudio de las características habituales del protocolo IEC 60870-5-104 y de la posterior generación de estadísticas y ratios necesarios para el lanzamiento de avisos por tráfico anómalo. Adicionalmente, se han implementado escenarios con los ataques más comunes, de forma que pueda validarse el sistema desarrollado.

Palabras clave

SCADA, IEC 60870-5-104, IDS

Abstract

The IEC 60870-5 family of standards defines the protocols commonly used for remote control in SCADA networks. These networks are responsible for the supervision, control and acquisition of relevant data in controlled industrial processes, such as those that are part of the energy production or treatment of resources for human benefit. Over the last five years, the number of threats to network security has increased exponentially, becoming one of the main problems for companies or organizations, both public and private. SCADA networks are not an exception, being of special importance the deployment of security measures that allow the proper functioning of the processes they control. The current trend towards the interconnection of most of the systems deployed, or the remote management of incidents, causes the appearance of multiple vulnerabilities, security failures and leaks in the data or resources of any entity. Specifically, this thesis studies security in the environment of an industrial system for supervision, control and data acquisition based on the IEC 60870-5-104 protocol, which is implemented in TCP/IP networks, and the realization of an IDS which can detect such intrusions in the network. This type of SCADA network is characterized by a low packet exchange rate and a very high regularity in the interval of requests to the server, which allows for the design of real-time threat detection solutions once the network characterization is achieved. The implemented IDS system is in charge of capturing the packets, analyzing these packets by studying the usual characteristics of the IEC 60870-5-104 protocol, and the subsequent generation of statistics and ratios required for the launch of anomalous traffic announcements. In addition, scenarios with the most common attacks have been implemented, so that the developed system can be validated.

Keywords

SCADA, IEC 60870-5-104, IDS

Agradecimientos

La entrega de este TFG supondrá un antes y un después en mi vida. Esta carrera y en especial el trabajo de fin de grado, me ha hecho darme cuenta del sacrificio que has de hacer para conseguir las cosas, y que en esta vida no hay nada que no puedas conseguir con trabajo, esfuerzo e ilusión.

En primer lugar, me gustaría agradecer este trabajo a mis padres, Marta y David, y mi hermano, Miguel, los cuales han conseguido siempre sacar lo máximo de mí, animándome y apoyándome en todo momento. Sé que durante el grado ha habido momentos duros, pero lo que nos ha hecho fuertes y me ha hecho crecer como persona es siempre el haber estado unidos. Gracias por inculcarme esta filosofía de vida y estar siempre a mi lado, por esa razón este trabajo no es mío, sino nuestro. También agradecer a toda la familia, que se ha preocupado e interesado por mí, y los cuales verán que todo este esfuerzo al final tiene su recompensa, y aquellas personas, que por razones de la vida no están aquí presentes, pero han estado a mí lado, ahí donde estéis, sé que también celebrareis este triunfo, tanto como yo o más. Sin olvidarme de mis padrinos, Mª José y Fernando, los cuales siempre me decían que una ingeniería es una carrera de fondo, que lo importante no es ir rápido, sino acabarla.

Agradecer a mi tutor, Jorge, por haberme propuesto este proyecto y sobre todo el interés e insistencia de vernos cada dos semanas para avanzar el proyecto y no dejarlo para última hora, gracias por explicarme las cosas, siempre que tenía dudas y con la paciencia que has tenido a lo largo del año. También agradecer a Sergio López Buedo y José Fernando Zazo, el interés que mostraron en este proyecto y la ayuda que me ofrecieron.

A lo largo de la vida, te cruzas con muchas personas, unas pasan de largo y otros dejan su huella, a veces son compañeros y otras rivales. Hay muchas personas que han dejado huella en mí. Ricardo Domingo, te conocí hace apenas 10 años, y desde entonces siempre has estado a mi lado, ya sea por estar en el equipo de fútbol, de compañero de carrera o ambas, pero tú también has hecho que esto sea posible. Sergio Vivas, puede que al principio fuésemos rivales, pero al final nuestros caminos se cruzaron y acabamos haciendo la misma carrera, estos últimos años has sido mi referencia en los estudios, siempre has conseguido lo que te has propuesto y nos has dejado el camino hecho para que nosotros sigamos tus pasos y sea más fácil. Antonio Campoy, puede que a ti te haya conocido hace relativamente poco, pero lo que has aportado en mí y el apoyo que has sido no tiene precio, el madrugar los 4 para los "DesayUAM" y el contar nuestro día a día y aconsejarnos es lo que me daba ánimos a seguir. Tampoco quiero olvidarme de las personas que he ido conociendo durante mi trayectoria, a los cuales puedo considerar como familia, siempre os estaré eternamente agradecidos por dejar esa huella en mí, Bea, Ester, Lucía, Rocío, Darío, Gisme, Gon, Javichu y Juampe.

Por último, agradecer a esas personas que han estado toda la vida a mi lado, y siempre se han ofrecido a ayudarme. Gracias a Alex y Dani, por estar siempre ahí y saber sacar lo mejor de mí. Gracias a todos los de "Téllez", Laura, Elena, Ana, Álvaro, Victoria. Aunque ahora nos separé la distancia por otros motivos, quiero agradeceros todo lo que habéis hecho por mí y deciros que os quiero por todo lo vivido.

ÍNDICE DE CONTENIDOS

1. INTRODUCCIÓN 1

1.1 MOTIVACIÓN 1
1.2 OBJETIVOS 1
1.3 FASES DE REALIZACIÓN 2
1.4 ORGANIZACIÓN DE LA MEMORIA 2

2. ESTADO DEL ARTE 5

2.1 INTRODUCCIÓN 5
2.2 SCADA 5
2.2.1 Dispositivos de Entrada/Salida (IO) 6
2.2.2 Unidades Terminales Remotas (RTU) 6
2.2.3 Controladores Lógicos Programables (PLC) 7
2.2.4 Unidades Terminales Maestras (MTU) 7
2.2.5 Interfaz Hombre-Máquina (HMI) 7
2.3 ESTÁNDAR IEC 60870-5-104 7
2.3.1 Arquitectura 9
2.3.2 Estructura 9
2.4 CIBERATAQUES 12
2.4.1 Stuxnet 13
2.4.2 Paquetes mal formados 13
2.4.3 Man in the middle 13
2.4.4 Ataque de predicción de número de secuencia TCP 13
2.5 SISTEMAS DE DETECCIÓN DE INTRUSIÓN 14
2.6 CONCLUSIÓN 14

3. DESARROLLO 15

3.1 INTRODUCCIÓN 15
3.2 ENTORNO DE DESARROLLO 15
3.2.1 VMware 15
3.2.2 QTester 104 de Ricardo L. Olsen 16
3.2.3 WinPP104 16
3.2.4 Entorno propio 18
3.3 SCAPY 20
3.4 ARQUITECTURA DESARROLLADA 21
3.4.1 Programación del detector 21
3.4.2 Programación de los ataques 23
3.5 NETWORKMINER 24
3.6 MEJORAS 25
3.7 CONCLUSIÓN 26

4. VALIDACIÓN 27

4.1 INTRODUCCIÓN 27
4.2 ATAQUES REALIZADOS 27
4.3 CASO REAL 34
4.4 CONCLUSIÓN 36

5. CONCLUSIONES Y TRABAJO FUTURO 37

5.1 CONCLUSIÓN 37
5.2 TRABAJO FUTURO 37

REFERENCIAS 39

ANEXOS I

A TIPOS DE IDENTIFICACIÓN I
B CAUSA DE TRANSMISIÓN III

ÍNDICE DE FIGURAS

FIGURA 1-1: EVOLUCIÓN DE LOS CIBERATAQUES EN ESPAÑA [11] 1
FIGURA 1-2: DIAGRAMA DE GANTT 2
FIGURA 2-1: ARQUITECTURA SCADA [1] 6
FIGURA 2-2: CAPAS MODELO OSI 8
FIGURA 2-3: ARQUITECTURA IEC 60870-5-104 9
FIGURA 2-4: CABECERA IEC 60870-5-104 [16] 10
FIGURA 2-5: CABECERA APCI I-FORMAT [16] 10
FIGURA 2-6: CABECERA APCI U-FORMAT [16] 10
FIGURA 2-7: CABECERA APCI S-FORMAT [16] 11
FIGURA 2-8: CABECERA ASDU [16] 12
FIGURA 3-1: ARQUITECTURA DEL ENTRONO 16
FIGURA 3-2: QTESTER 16
FIGURA 3-3: CONFIGURACIÓN DEL PROGRAMA WINPP104 17
FIGURA 3-4: WINPP104 17
FIGURA 3-5: CLASE DE PAQUETES CON FORMATO DE SUPERVISIÓN 18
FIGURA 3-6: CABECERA ASDU PARA PAQUETES CON TRANSFERENCIA DE DATOS DEL TIPO 45 18
FIGURA 3-7: CÓDIGO DE ESTABLECIMIENTO DE UNA CONEXIÓN 19
FIGURA 3-8: EJEMPLO DE CREACIÓN DE PAQUETES DE IEC 60870-5-104 19
FIGURA 3-9: *MAIN* DEL ENTORNO CREADO POR EL ALUMNO 20
FIGURA 3-10: CÓDIGO DE LA LIBRERÍA DE PAQUETES DE TRANSFERENCIA DE DATOS 23
FIGURA 3-11: CÓDIGO DEL FORMATO TRANSFERENCIA DE DATOS CON *TYPEID* 45 23
FIGURA 3-12: CÓDIGO DE LA IMPLEMENTACIÓN DE LA LISTA DE PAQUETES 24
FIGURA 3-13: CAMBIAR LA DIRECCIÓN IP 24
FIGURA 3-14: NETWORKMINER 25
FIGURA 4-1: PRIMER CASO 28
FIGURA 4-2: NETWORKMINER PRIMER CASO 28
FIGURA 4-3: SEGUNDO CASO 29
FIGURA 4-4: *NETWORKMINER* SEGUNDO CASO 29
FIGURA 4-5: TERCER CASO 30
FIGURA 4-6: *NETWORKMINER* TERCER CASO 30
FIGURA 4-7: CUARTO CASO 31
FIGURA 4-8: *NETWORKMINER* CUARTO CASO 31
FIGURA 4-9: WIRESHARK CUARTO CASO 32
FIGURA 4-10: WARNING 33
FIGURA 4-11: TRAZA REAL DE INTERNET 33
FIGURA 4-12: NETWORKMINER CAPTURA DE INTERNET 34
FIGURA 4-13: CASO REAL 36

ÍNDICE DE TABLAS

TABLA 2-1: TIPOS DE ASDU 12

GLOSARIO

ASDU *Application Service Data Units* (Unidades de Datos de Servicios de Aplicación), bloques de datos de la capa de IEC 60870-5-104.

APDU *Application Protocol Data Units* (Unidades de Datos de Protocolos de Aplicación), cabecera de IEC 60870-5-104.

IEC *International Electrotechnical Commission* (Comisión Internacional Electrotécnica), organismo que se encarga de establecer unas bases en el campo eléctrico, electrotécnico y tecnologías relacionadas.

HPCN *High Performance Computing and Networking* (Grupo de Investigación de Redes y Compitaciones de Altas Prestaciones).

SCADA *Supervisory Control And Data Acquisition* (Supervisión, Control y Adquisición de Datos). Sistemas de comunicaciones que permiten controlar procesos a distancia.

IP *Internet Protocol* (Protocolo de Internet) se da en la capa de Red.

TCP *Transmission Control Protocol* (Protocolo de Control de Transmisión) se da en la capa de transporte.

HMI *Human-Machine Interface* (Interfaz Humano Máquina), software que facilita al humano la toma de decisiones.

PLC *Programmable Logic Controllers* (Controlador Lógico Programable), dispositivo con la función de automatizar los procesos electromecánicos.

RTU *Remote Terminal Unit* (Unidad Terminal Remota), dispositivo encargado de obtener la información y distribuirla a otra unidad remota.

OSI *Open System Interconnection* Interconexión de sistemas abiertos.

ISO *International Organization for Standardization* Estándar internacional de organización.

Industria 4.0 Término para referirse a la nueva revolución industrial.

IOT *Internet Of Things* (Internet de las Cosas). Concepto que se utiliza para decir que los objetos cotidianos se conectan a Internet.

1. Introducción

1.1 Motivación

El número de ciberataques que se han producido a lo largo de los últimos años ha seguido una tendencia exponencial, Figura 1-1, causando robo de identidades, fraudes, manipulación de datos y reprogramación de máquinas. Todo ello hace que el ser humano desconfíe de Internet, siendo cada vez más importante vigilar y tener un control de todas nuestras conexiones.

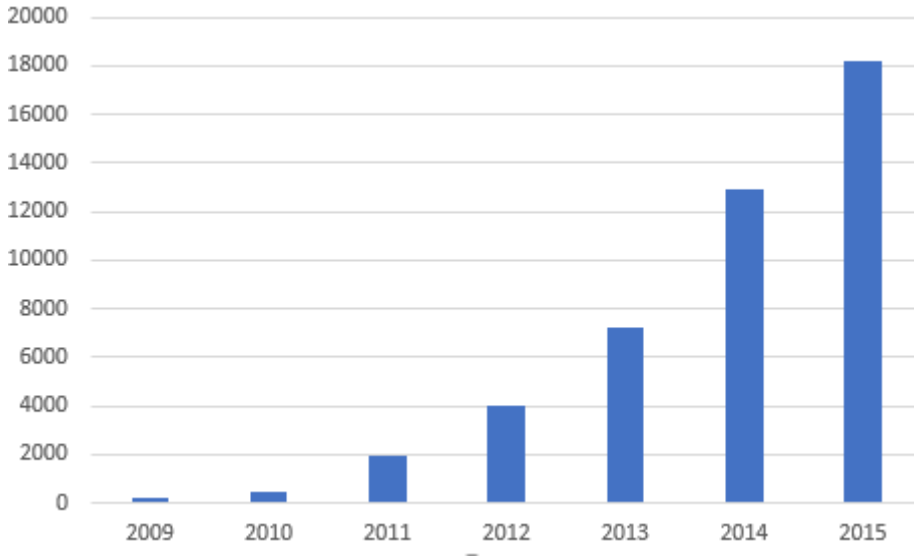


Figura 1-1: Evolución de los ciberataques en España [12]

El concepto de industria 4.0, es uno de los temas más hablados hoy en día. Consiste en una nueva revolución que combina las técnicas de producción con las tecnologías inteligentes que se integrarán en las organizaciones, personas y activos. Esta revolución se ve marcada por la aparición de nuevas tecnologías como es el Internet de las Cosas (*IOT*) y la automatización de todos los procesos, por ello, la seguridad será un campo importante y necesario de investigar, creando controles sobre los protocolos que llevan dichos procesos.

1.2 Objetivos

El objetivo de este Trabajo Fin de Grado ha sido el desarrollo de un sistema de detección de tráfico anómalo en redes SCADA (*Supervisory Control And Data Acquisition*) basadas en el protocolo IEC 60870-5-104 [1]. Lo que queremos lograr con este programa IDS (*Intrusion Detection System*), es analizar el tráfico de la red en tiempo real y ser capaz de detectar cuando un paquete es anómalo.

En la implementación del sistema de detección, primero tenemos que estudiar los campos del protocolo, seguido de un análisis de los tipos de ataques se pueden realizar, estableciendo una serie de normas que deberán cumplir los paquetes para no ser anómalos

y poder certificar que es una conexión segura. Dicho IDS tiene como objetivo que sea capaz de identificar cuando se produce una anomalía y que el tiempo de ejecución no sea extremadamente elevado.

1.3 Fases de realización

En la realización de este proyecto, se tuvo que llevar una serie de tareas que se detallan a continuación y en la Figura 1-2:

- **Estudio del estado del arte**: Esta fase ha sido la de más larga duración, ya que se tenía que estudiar con profundidad cómo funcionaban las redes SCADA y en especial el protocolo IEC 60870-5-104, tanto su estructura como los diferentes tipos de formatos. Además, se tuvo que indagar en temas desconocidos, cómo son los diferentes ataques que se podrían producir en dichos protocolos.

- **Desarrollo**: En esta etapa se implementaron distintos programas que analizan las anomalías sobre el protocolo IEC 60870-5-104, en el cual se tuvo que familiarizar con la librería Scapy de Python.

- **Validación**: Durante esta fase se pudo demostrar que lo que se había programado funcionaba. Corrigiendo algunos fallos que se tenían en el código hasta perfeccionarlo.

- **Memoria**: En dicha fase se redactó un documento que reflejaba el aprendizaje, para poder facilitar y transmitir, a la vez, a personas venideras el poder partir de una base que ayudara a entender mejor, y progresar en el trabajo alcanzado.

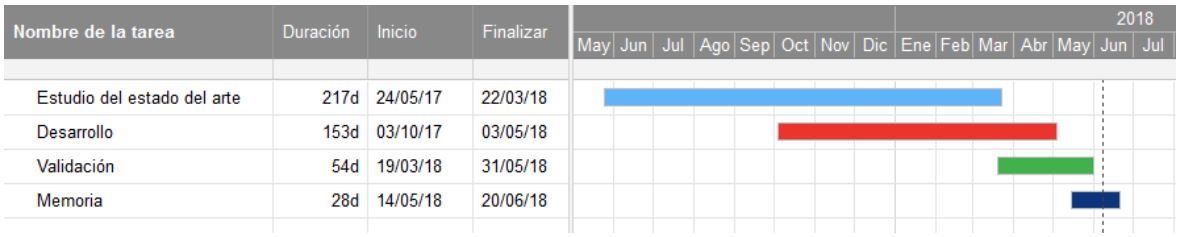


Figura 1-2: Diagrama de Gantt

1.4 Organización de la memoria

La demostración de los resultados alcanzados a lo largo del trabajo en la memoria se puede ver en los siguientes capítulos:

- **Capítulo 2: Estado del arte.** En este capítulo se introducirá el tema del que trata el trabajo y, a su vez, se dará un enfoque básico de los conocimientos necesarios para poder comprender el proyecto sin dificultad. Se hablará de las redes SCADA y de

los dispositivos que componen dicha red. Una vez introducido ese tema, se detallará todo lo necesario para entender el protocolo IEC 60870-5-104, se explicarán los posibles ataques que se pueden dar en dicho protocolo. Finalizando el capítulo con la explicación de un IDS y el porqué será necesario.

- **Capítulo 3: Desarrollo.** En este capítulo se detallan los aspectos que han sido necesarios en la realización del trabajo y su implementación. Al principio, se explicará la razón de la necesidad de las máquinas virtuales en este proyecto y los dos programas base utilizados, creando un entorno con conexiones de nuestro protocolo a analizar. Se explicará cómo instalar la librería de Scapy y qué funciones se han utilizado en la implementación del detector. Como conclusión, se reflejarán las pautas implementadas en el IDS, el cual debe detectar todas las anomalías que se produjesen.

- **Capítulo 4: Validación.** Este capítulo incluye los resultados de las pruebas que se realizaron a lo largo del desarrollo de todo el proyecto. También se analizará un caso real en el cual la Cátedra UAM-Naudit HPCN facilitó el acceso a una traza con este protocolo, demostrando la utilidad que puede tener este Trabajo Fin de Grado en una empresa real.

- **Capítulo 5: Conclusiones y trabajo futuro.** En este capítulo se concluirá dicho trabajo y se darán las razones por las cuales se decidió escoger este proyecto, la relación que tiene con el grado y lo que puede aportar. Finalmente, se comenta una serie de mejoras que se podrían implementar en un futuro cercano para dicho detector.

2. Estado del arte

2.1 Introducción

En este capítulo se expondrá el estado del arte relativo a las redes SCADA, los dispositivos que se utilizan en dichas redes, el protocolo IEC 60870-5-104, distintos ciberataques y sistemas de intrusiones, todo ello relacionado con la realización de este Trabajo Fin de Grado.

El objetivo de este capítulo es dar una noción al lector para que se familiarice con el tema haciéndolo más fácil de entender.

2.2 SCADA

Los sistemas SCADA constan de una parte de software y otra de hardware que permiten a las industrias organizarse en los siguientes aspectos:

- Control de procesos industriales, tanto localmente como remotamente.
- Supervisar, recopilar y procesar datos en tiempo real.
- Interactuar directamente con dispositivos como sensores, válvulas, bombas, motores a través del HMI (*Human Machine Interface*).
- Recopilar datos en un archivo de registro (*log*).

Los sistemas SCADA se han vuelto imprescindibles en el sector industrial. Con su ayuda obtenemos una mayor eficiencia; procesan los datos y gracias a ellos podemos tomar mejores decisiones; detectan fácilmente los problemas, y esto nos permite una rápida solución de los mismo para poder así evitar las caídas del sistema.

La arquitectura básica de un sistema SCADA empieza por un PLC (*Programmable Logic Controller*, Controlador Lógico Programable) o una RTU (*Remote Terminal Unit*, Unidad Terminal Remota). Estos dispositivos son unos microordenadores que se comunican con un *array* de sensores y luego enrutan esa información a ordenadores con software SCADA. Este software se encarga de procesar, distribuir y mostrar los datos. Un ejemplo de arquitectura sería el visto en la Figura 2-1.

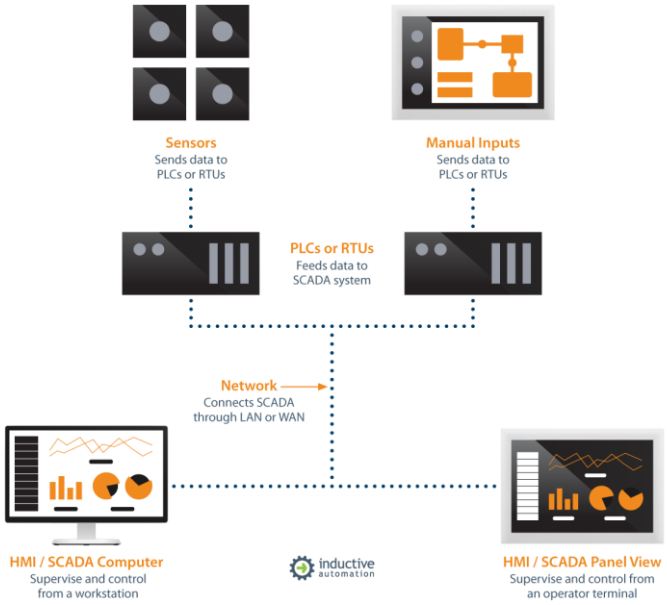


Figura 2-1: Arquitectura SCADA [2]

Los sistemas SCADA son frecuentes en muchos sectores industriales: agua, petróleo, transporte, energía y muchos más. Este trabajo se centrará en el protocolo IEC 60870-5-104 y será el que analizaremos en este trabajo. [2]

A continuación, se dará una breve introducción de los dispositivos que forman las redes SCADA y cuáles son sus funciones principales.

2.2.1 Dispositivos de Entrada/Salida (IO)

Los dispositivos de Entrada/Salida (IO) son instrumentos como sensores, actuadores o dispositivos de control. Estos dispositivos están conectados de tal forma que captan la información de las señales para así poder medirlas. Las señales que captan pueden ser analógicas o digitales. Cuando nos queremos referir a la medida de instrumentos utilizaremos señales analógicas. En cambio, si queremos medir pulsos de flujo, el estado de un dispositivo o los interruptores de un detector utilizaremos señales digitales.

2.2.2 Unidades Terminales Remotas (RTU)

Las unidades terminales remotas están distribuidas por toda la arquitectura y su función principal es la de recolectar la información de los dispositivos IO y convertir los datos en digitales. A continuación, los datos se envían al sistema de supervisión para su procesamiento. En comparación con los PLC, las RTU son más adecuadas en redes inalámbricas. Estos dispositivos en la arquitectura SCADA serían los denominados esclavos.

2.2.3 Controladores Lógicos Programables (PLC)

Los controladores lógicos programables se conectan a los dispositivos IO para obtener señales en tiempo real, y después transferirlas como datos digitales a los sistemas de supervisión. Estos dispositivos son más comunes en plantas de control de área local, a diferencia de las RTUs. Otra gran diferencia entre estos dos dispositivos es el tamaño y la capacidad: los RTU normalmente tienen más entradas y salidas que un PLC. Además, la configuración necesaria de un PLC suele requerir una programación extra o un script para poder procesar cualquier tarea lógica. Estos dispositivos, al igual que los RTUs, en un sistema SCADA tendrían una función de esclavos.

2.2.4 Unidades Terminales Maestras (MTU)

Las unidades terminales maestras en un sistema SCADA son unos dispositivos que se encargan de enviar los comandos para las RTUs y PLCs, los cuales están localizados en sitios remotos desde el control, recolectan los datos requeridos, almacenan y procesan la información para poder mostrarla en forma de imágenes, gráficas o tablas en formato HMI. Gracias a estos dispositivos es más fácil la toma de decisiones. La comunicación entre los dispositivos MTU y PLC o RTU, es bidireccional. La única diferencia es que los dispositivos que actúan como esclavos, RTUs y PLCs, no pueden iniciar una conexión, simplemente recolectan información y se la proporcionan después a las MTUs. En el sistema SCADA las MTUs actúan como el corazón del sistema.

2.2.5 Interfaz Hombre-Máquina (HMI)

La interfaz hombre-máquina es un software que proporciona a los operadores humanos una manera de interactuar y manejar un sistema. Esta iteración es a través de una interfaz gráfica de usuario (GUI), que facilita la información intercambiada y la comunicación entre dos tipos de HMIs: Nivel de supervisión y nivel de máquina. El software de HMI está diseñado para ambos niveles. Gracias a este software, el humano es capaz de tomar mejores decisiones con los datos obtenidos.

2.3 Estándar IEC 60870-5-104

Como se explica en la página de IPComm [3], el protocolo IEC 60870-5-104 es un estándar internacional implantado por el IEC desde el año 2000. Dicho protocolo está basado en el protocolo 60870-5-101, y por eso tienen la misma capa de aplicación.

Este protocolo permite la comunicación entre la MTU y los distintos PLCs o RTUs. Para ello se utiliza conexiones a través de TCP/IP. El protocolo TCP garantiza una transmisión de datos fiable. Además, por norma, este protocolo tiene establecido que el puerto estándar sea el 2404 de TCP.

El protocolo IEC 60870-5-104 posee las mismas ventajas y desventajas que el IEC 60870-5-101, su principal ventaja es que permite la conexión a través de una red estándar, lo que permite una transmisión de datos entre varios dispositivos.

El protocolo del 104 cumple un modelo ISO/OSI con la estructura de la Figura 2-2.

<table>
<tr><td>7. Aplicación</td><td rowspan="3">ASDU</td></tr>
<tr><td>6. Presentación</td></tr>
<tr><td>5. Sesión</td></tr>
<tr><td>4. Transporte</td><td>TCP</td></tr>
<tr><td>3. Red</td><td>IP</td></tr>
<tr><td>2. Enlace</td><td rowspan="2">ETHERNET/
Acceso a las subredes</td></tr>
<tr><td>1. Física</td></tr>
</table>

Figura 2-2: Capas Modelo OSI

- Nivel 1: El nivel físico se encarga de la transmisión física de ceros y unos lógicos que forman la traza.

- Nivel 2: El nivel de enlace se encarga de transmitir trazas sin errores entre equipos conectados en la misma red física.

- Nivel 3: El nivel de red es el encargado de transmitir las trazas a equipos que no están conectados en la misma red física.

- Nivel 4: El nivel de transporte se ocupa de transportar los mensajes de la capa de aplicación entre los terminales de la aplicación.

- Nivel 5: El nivel de sesión permite delimitar y sincronizar el intercambio de datos, incluyendo los medios para crear un punto de restauración y un esquema de recuperación.

- Nivel 6: El nivel de presentación proporciona servicios que permiten a las aplicaciones que se comunican interpretar el significado de los datos intercambiados.

- Nivel 7: El nivel de aplicación es donde residen las aplicaciones de red y sus protocolos.

2.3.1 Arquitectura

El protocolo 104 tiene una jerarquía muy clara que se divide en dos grupos: los maestros y los esclavos. El maestro manda de manera continuada tramas a los esclavos, bien para dar órdenes o para recopilar información. En cambio, el esclavo sólo puede contestar cuando el maestro le pregunta. La Figura 2-3 representaría la arquitectura más simple que se puede dar en este protocolo.

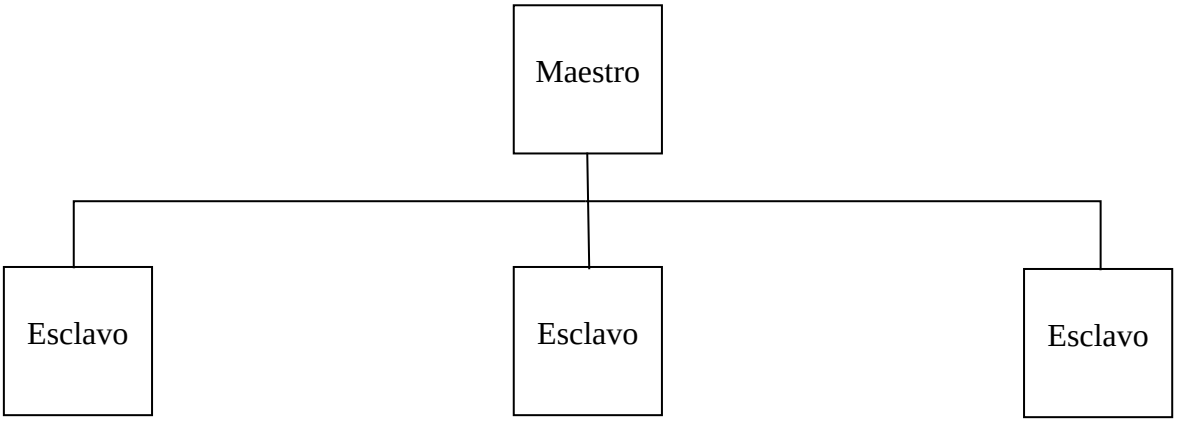


Figura 2-3: Arquitectura IEC 60870-5-104

2.3.2 Estructura

En la Figura 2-4 se muestra la organización de la trama del protocolo. Según la Norma la trama recibe el nombre de APDU y se puede distinguir dos campos en los que se divide. que todos los paquetes tienen en común los campos de START, que al ser el protocolo IEC 60870-5-104 siempre tendrá el valor de 104 en decimal, y el campo que determina la longitud de la cabecera de APDU. Los demás campos pueden ir variando dependiendo de la función de dicho paquete.

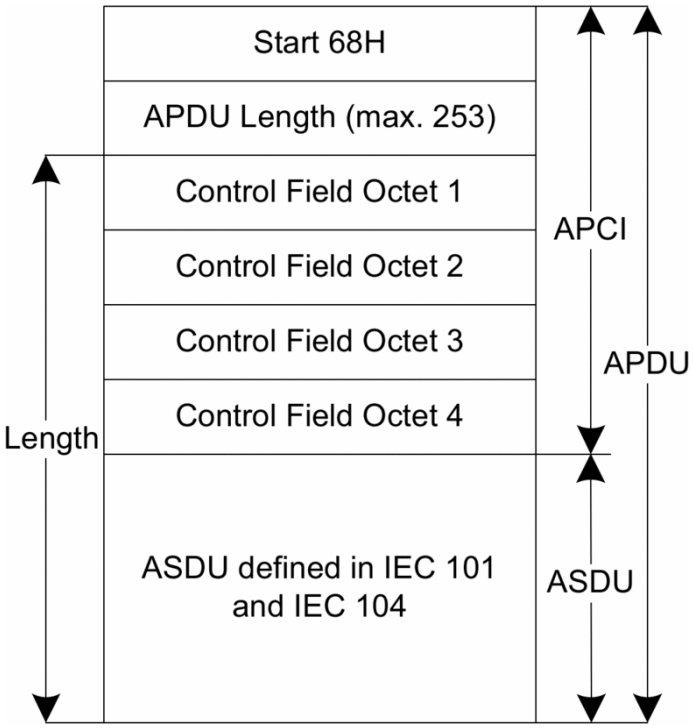


Figura 2-4: Cabecera IEC 60870-5-104 [17]

byte\bit	7	6	5	4	3	2	1	0
0	Send sequence number N(S) LSB							0
1	Send sequence number N(S) MSB							
2	Receive sequence number N(R) LSB							0
3	Receive sequence number N(R) MSB							

Figura 2-5: Cabecera APCI I-format [17]

byte\bit	7	6	5	4	3	2	1	0
0	TESTFR		STOPDT		STARTDT		1	1
1	0							
2	0							
3	0							

Figura 2-6: Cabecera APCI U-format [17]

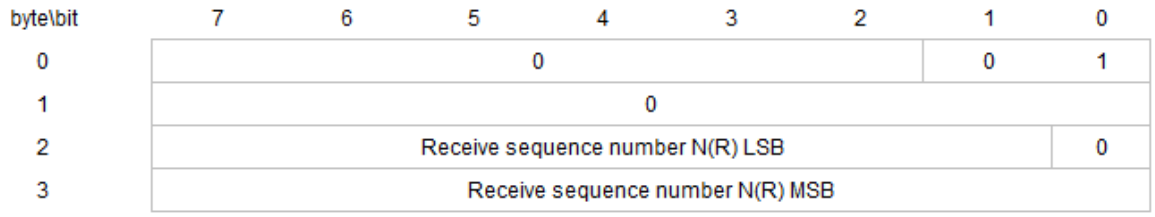


Figura 2-7: Cabecera APCI S-format [17]

Como podemos observar en las figuras de las cabeceras APCI podemos identificar cada tipo de formato con los dos últimos bits menos significativos del primer Byte:

- Si es b00, será formato de transferencia de información (*Information transfer*, I-format) (Figura 2-5).
- Si es b01, será formato supervisado numerado (*numbered Supervisory*, S-format), (Figura 2-7).
- Si es b11, será formato no numerado (*Unnembered control format*, U-format), (Figura 2-6).

Una vez identificados los formatos, si es U-format podemos distinguir 3 tipos:

- STARDT (*START Data Transfer*) comenzar una conexión nueva.
- STOPDT (*STOP Data Transfer*) cerrar la conexión.
- TESTFR (*TEST FRame*) probar si la conexión sigue activa.

Los paquetes S-format y U-format, tienen una longitud fija de valor 4. En cambio, en los I-format la longitud es variable. Además, si es I-format, el paquete tendrá una cabecera adicional, la cabecera ASDU, que se puede diseccionar en dos partes: el identificador de la unidad de datos y la información del objeto. Dicha cabecera está compuesta por los campos:

- *Type Identification* (TypeID). Dependiendo del valor de este campo sabremos a qué se destina dicho paquete, como podemos ver en la Tabla 2-1 y en el anexo A.A. Toma valores entre 0 y 255 (7 bits). Aquellos valores que no aparecen en la tabla, o bien no se utlizan o están reservados para casos especiales.
- *Variable Sructure Qualifier* (SQ). Este campo especifica si el objeto tendrá información (1 bit).
- *Cause of Transmission* (CoT) . El valor de este campo determinará a dónde dirigir el ASDU para su tarea de aplicación específica (6 bits).
- *Common Address*. La dirección común está asociada a todos los objetos ASDU. La dirección global es la de *Broadcast*, dirigida a todos los dispositivos de un campo específico (2 Bytes).
- *Information Object Address*. La dirección de la información del objeto es usada como dirección de destino en el control de dirección y como dirección origen en la dirección de monitorización (3 Bytes).

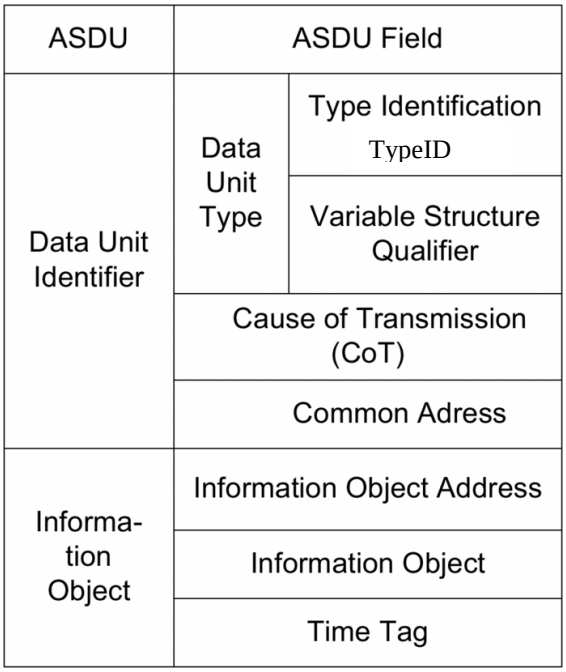


Figura 2-8: Cabecera ASDU [17]

Tabla 2-1: Tipos de ASDU

TypeID	
1-44	Información de proceso en dirección de monitorización (de esclavo a maestro)
45-69	Información de proceso en la dirección de control (de maestro a esclavo)
70-99	Información del sistema en la dirección de monitorización
100-109	Información del sistema en la dirección de control
110-119	Parámetros en la dirección de control
120-127	Transferencia de archivos

Una vez, hecho el estudio sobre como funcionan las redes SCADA y el protocolo IEC 60870-5-104, se explicarán los diferentes ataques que se pueden dar en dichas redes, así como un ejemplo real.

2.4 Ciberataques

Internet se ha convertido en una herramienta crítica para muchas instituciones actuales. Muchas personas también confían en Internet para llevar a cabo muchas de sus actividades profesionales. Pero detrás de todas estas actividades, hay un lado oscuro, un lado donde ciertas personas buscan la manera de hacer daño a través de ordenadores.

Dada la frecuencia y variedad de ataques existentes, así como la amenaza de nuevos, la ciberseguridad se ha convertido en un tema principal en el campo de las redes. En esta sección veremos un ejemplo real de un ataque, y los distintos ataques que se pueden dar en el protocolo del 104.

2.4.1 Stuxnet

Stuxnet fue un gusano informático que afectaba a los equipos con el sistema operativo de Windows. Stuxnet era capaz de propagarse con suma facilidad entre los equipos con el mismo sistema operativo, sin necesidad de conexión a Internet y sin dejar huella. La funcionalidad de dicho malware era infectar las centrifugadoras de uranio del programa nuclear iraní, y hacer que funcionaran de forma anómala para que se rompieran. Aún no se ha descubierto quién fue el culpable de la creación de dicho virus, pero sí que se sabe que no se realizó con fines económicos, sino con fines políticos, pues querían hacer desaparecer una planta de enriquecimiento de uranio en Irán. [9]

Este hecho fue el cambio por el cual se le empezó a dar más importancia al campo de la ciberseguridad. En las subsecciones venideras se explicarán algunos de los posibles ciberataques que se pueden dar en el protocolo objeto de estudio.

2.4.2 Paquetes mal formados

Enviar paquetes malformados puede producir averías en el sistema. Por lo que para este proyecto se estudia los distintos campos, en especial del protocolo del 104, en el cual se comprueban que los valores sean correctos y poder evitar dicho ataque. [13]

2.4.3 Man in the middle

Man in the middle u "Hombre en el medio" (ataque por interposición) es difícil de identificar y de defenderlo. La idea que hay detrás de este ataque es interponerse en la comunicación entre el remitente y el destinatario, y poder acceder al tráfico, modificarlo y reenviarlo modificado al otro extremo. [14]

2.4.4 Ataque de predicción de número de secuencia TCP

El cliente a cada lado de la sesión TCP mantiene un número de secuencia de un tamaño de 4 bytes, que utiliza para realizar un seguimiento de la cantidad de datos que envía. Este número de secuencia se incluye en cada paquete transmitido y el receptor lo reconoce como número de asentimiento al emisor que los datos se han transmitido correctamente.

Al iniciar una conexión sobre TCP dicho número de secuencia toma un valor inicial aleatorio y se va incrementando a medida que la conexión avanza.
El ataque de predicción del número de secuencia de TCP se hace para intentar hacerse pasar por otra persona. Si un atacante consiguiera adivinar el patrón por el cual los números de secuencia se van modificando, sería capaz de poder averiguar cuál serían los siguientes. Si se consiguen esos números de secuencia, el atacante podría meterse en medio de la conexión y hacerse pasar por uno de los integrantes, y enviar mensajes a los demás robándoles su identidad. Dicho ataque se puede dar en el protocolo que se investiga, ya que se transporta por TCP. [11]

Ahora que se conocen los distintos ataques que se pueden dar en estas redes, se explicaran los sistemas son capaces de detectarlos. Dependiendo de sus funciones se hará una distinción entre ellos.

2.5 Sistemas de detección de intrusión

La función principal de un sistema de intrusión es el análisis de la red, con el fin de poder detectar comportamientos sospechosos de los dispositivos conectados y analizar los distintos campos de los paquetes buscando anomalías. Podemos distinguir dos tipos de sistemas de intrusiones. El primero sería un sistema IDS: este tipo de sistema sólo se encarga de detectar los ataques y notificarlo. En cambio, un sistema IPS (*Intrusion Protection System*) tiene la misma funcionalidad que un IDS, pero puede detener el ataque. La mejor implementación de un buen IDS sería aquella que tuviera una eficacia del 100% y además fuera lo más eficiente posible.

Una vez explicados los IDS podemos diferencias dos familias: N-IDS (*Network-Intrusion Detection System*) y H-IDS (*Host-Intrusion Detection System*). La primera familia se encarga de escuchar y capturar el tráfico en la red, de forma que ningún usuario pueda detectarlo, siendo capaz de analizar una red con varios dispositivos. En este tipo de sistemas es necesario un hardware exclusivo. En cambio, los sistemas H-IDS están integrados dentro de un host, por lo que se encargan de analizar el tráfico del propio host, no obstante, también puede centrarse en otras fuentes de información, tales como el comportamiento del equipo (procesos ejecutándose, consumos de memoria y CPU, o, sobre todo, que archivos se han modificados). [5]

2.6 Conclusión

En esta sección han quedado definidos las redes SCADA, y los dispositivos que la forman, una introducción del protocolo IEC 60870-5-104, así como su jerarquía y la estructura con la definición de todos los campos del protocolo. Se han explicado también los distintos ataques que se pueden dar, seguido de los sistemas que son capaces de detectarlos. Este capítulo nos ayudará a entender como diseñar el disector de manera correcta y entender su funcionamiento.

3. Desarrollo

3.1 Introducción

En este capítulo se explican las distintas fases llevadas a cabo en la implementación del IDS sobre el protocolo IEC 60870-5-104. Se comienza describiendo el entorno de desarrollo que se ha utilizado y su configuración. A continuación, se detallan los programas que han sido utilizados en el establecimiento de dicha conexión. Más adelante, se hace hincapié en las distintas funciones de la librería Scapy implementadas en la programación del proyecto y las distintas normas que se han impuesto a la hora de realizar el diseño del detector. Así como, una breve explicación de como se han hecho los diferentes ataques a lo largo del proyecto.

3.2 Entorno de desarrollo

El primer paso para poder realizar pruebas sobre nuestro detector, consiste en crear un entorno de desarrollo, en el cual se puedan establecer conexiones para el protocolo IEC 60870-5-104. Habiendo utilizado la máquina virtual VMware [22], debido a la sencillez en su manejo, se llegó a la conclusión de que instalar un entorno de desarrollo sobre esta herramienta, agilizaría el proceso de diseño.

3.2.1 VMware

Para la configuración del entorno se crean dos máquinas virtuales, con un sistema operativo de Windows 7. Una de ellas se encarga de llevar a cabo la ejecución del programa *QTester104 de Ricardo L. Olsen [20]*, mientras la otra ejecuta una demo de *WinPP104 [21]*.

Para poder establecer una conexión entre ambas máquinas y que se consiga una correcta coordinación entre ambas, se lleva a cabo un proceso de configuración previo de ambas máquinas virtuales. Para este proceso es necesario ir a la pestaña de configuración y en la sección de red se debe modificar el tipo a modo NAT o bien, a modo BRIDGE. Una vez se configure el tipo de conexión de la máquina virtual pueden abrirse ambas máquinas virtuales y para crear la conexión entre ellas, además, se puede abrir una tercera que actúe como IDS para llevar a cabo un análisis del tráfico.

En la Figura 3-1 se muestra cómo será la arquitectura que se configura, para la realización de este proyecto. Una de las máquinas virtuales será el maestro, otra el esclavo y una tercera que actuará de IDS implementado en este trabajo.

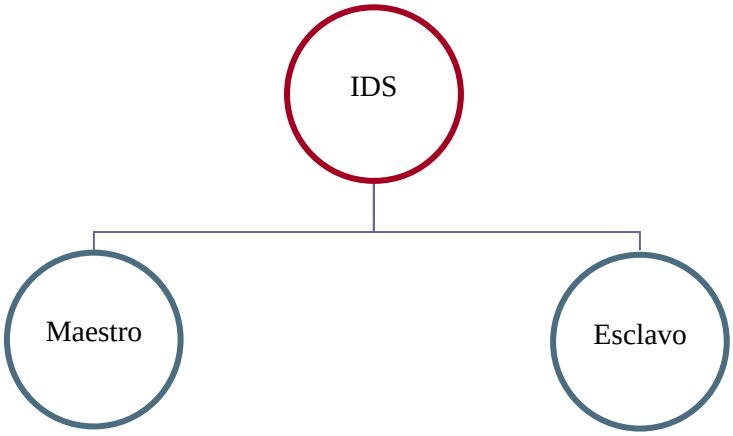


Figura 3-1: Arquitectura del entrono

3.2.2 QTester 104 de Ricardo L. Olsen

Este programa es muy intuitivo de utilizar, tal y como se puede ver en la Figura 3-2. Se aprecian varios campos con los cuales, existe la posibilidad de modificar la configuración del *QTester 104* fácilmente. El campo *Remote IP Address* se rellena con la dirección IP del ordenador con el que se desea establecer conexión. A continuación, el campo *Command Type* determinará qué tipos de paquetes se desean transmitir. Al ser un programa gratuito, tiene limitaciones en cuanto a tipos de paquetes a transmitir, por este motivo sólo hay opción de escoger entre una variedad de 13 opciones. Además, dicho programa deja marcar las pestañas *Log Messages* y *Autoscroll* para que se vayan mostrando por pantalla, todos los mensajes de la conexión. Este programa será el encargado de mantener el rol de maestro en el entorno que durante este proyecto se desarrolla.

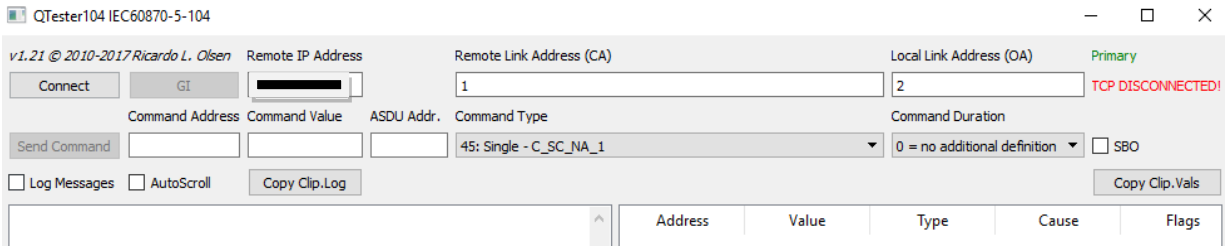


Figura 3-2: QTester

3.2.3 WinPP104

La versión analizada debido a su categoría de demo tiene una serie de limitaciones que repercuten en la recepción de paquetes. A la hora de realizar la configuración del programa, deberemos seleccionar la pestaña *Parameterize*, seleccionando la opción *Receiver/Transmitter1…*, podremos configurar a nuestro gusto las siguientes opciones, tal y como se muestra en la Figura 3-3:

1. El campo de *Function* debe de ser *Substation = TCP server*
2. Configuración las direcciones IP correspondientes
3. Establecer el *Port number RTU (Server)* con valor 2404

Este programa será el encargado de llevar a cabo las funciones de Esclavo en el desarrollo del trabajo.

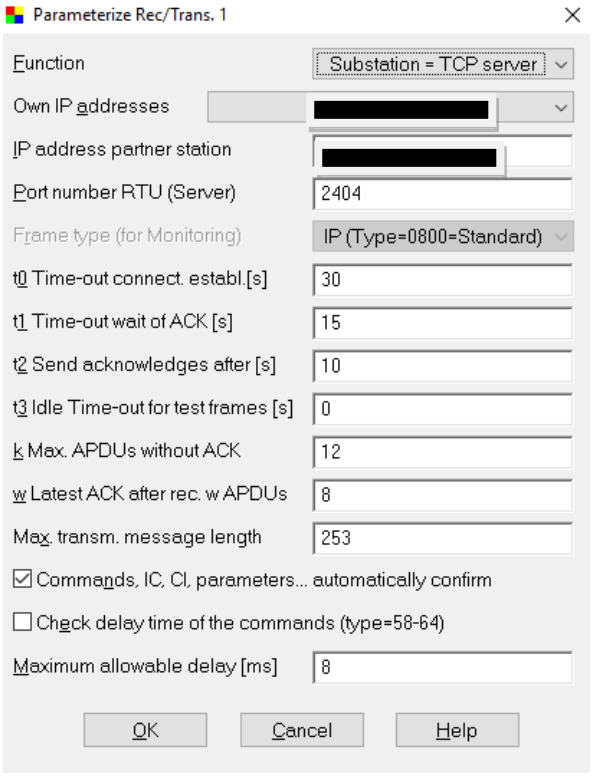


Figura 3-3: Configuración del programa WinPP104

Así mismo, este programa facilita un contador de paquetes que recibe y transmite mostrando un campo que permite saber si se ha producido algún error durante la conexión. Este software es de gran utilidad a la hora de comprender como funciona una conexión que utiliza el protocolo IEC 60870-5-104, aunque, como se ha comentado anteriormente, con una limitación de recepción de paquetes. Una vez se alcanza dicho límite, emerge un recordatorio de que se utiliza una demo, debiéndose reiniciar el programa para seguir usándolo, tal y como se detalla Figura 3-4.

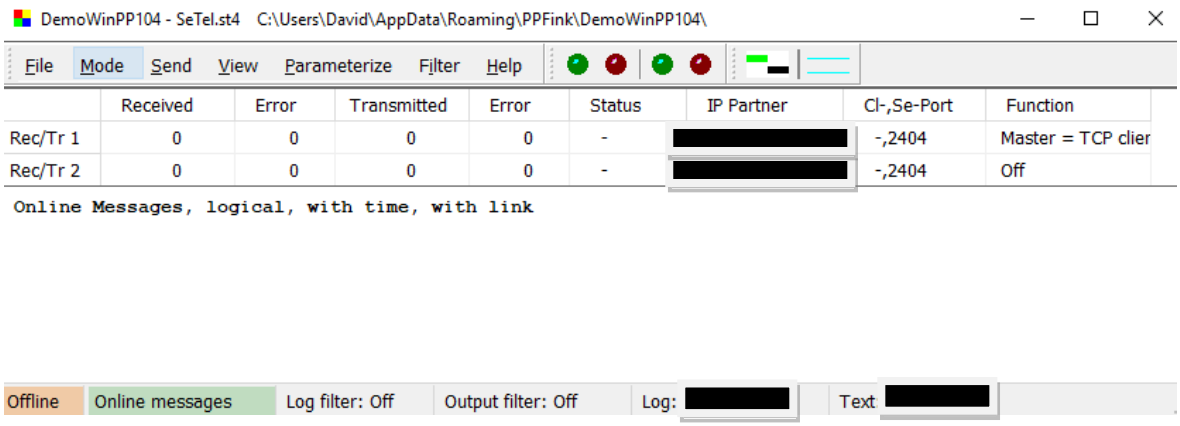


Figura 3-4: WinPP104

3.2.4 Entorno propio

Ante la inexistencia de un programa que permitiera la modificación de campos con generación de anomalías, se opta por el desarrollo de un entorno propio con Python capaz de conectar con un esclavo y poder intercambiar paquetes con anomalías.

El primer paso consiste en la programación de una librería, la cual establece el tamaño necesario para poder generar paquetes permitiendo la modificación de campos. Esta tarea fue una de las más complicadas del proyecto, ya que el alumno nunca había pensado que se podría dar dicha situación, aun así, gracias al tutor y a la capacidad resolutiva del alumno, este contratiempo se solventó. Más adelante en la sección de trabajo futuro, se comentarán los aspectos en los que dicho entorno se podría mejorar.

Durante el desarrollo de esta sección se lleva a cabo el planteamiento de varios archivos, entre ellos la librería de la cual se muestra un ejemplo en la Figura 3-5. Es aquí cuando se presenta un ejemplo de la programación llevada a cabo para cada uno de los formatos y en sucesión una serie de cabeceras ASDU con distintos *TypeID* como se observa en la Figura 3-6.

```
class s_frame(Packet):
      name = "s_frame"
      fields_desc = [ XByteField("START", 0x68),
                  XByteField("ApduLen", 0x04),
                  LEShortField("Type", 0x01),
                  LEShortField("Rx", 0x0000)]
```

Figura 3-5: Clase de paquetes con formato de supervisión

```
class asdu_infobj_45(Packet):
      name = "C_SC_NA_1"
      fields_desc = [
                  X3BytesField  ("IOA", 0x23),
                  XByteField("SCO", 0x80)]
```

Figura 3-6: Cabecera ASDU para paquetes con transferencia de datos del tipo 45

Una vez establecida esta base deberá implementarse la apertura de una conexión nueva sobre TCP/IP, haciendo que ésta se mantenga abierta. Se programa un código capaz de abrir un *socket* y dar comienzo a la conexión, configurándolo de tal manera que se estableciera ya una conexión sobre TCP enviando los paquetes de *Handshake three way*, como se muestra en la Figura 3-7, dicho código se programa para que se conecte al puerto genérico del protocolo IEC 60870-5-104.

```
HOST = ''
PORT = 2404
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.setsockopt(socket.SOL_SOCKET,socket.SO_LINGER,struct.pack('ii', 1, 0))
#s.settimeout(1)
s.bind((HOST, PORT))
s.listen(10)
```

Figura 3-7: Código de establecimiento de una conexión

Ahora que ya se tiene la librería que permite establecer la longitud de los campos necesarios para crear paquetes y ya se ha implementado el código que abre y conecta los equipos, el siguiente paso que se debe dar es instaurar los paquetes del protocolo IEC 60870-5-104 que se quieran transmitir. Para ello se crea una lista con todos los paquetes que se desean enviar. Dicha lista está implementada de tal forma que debe seguir el orden de campos que se presenta a continuación:

1. *START*
2. *ApduLen*
3. *Type,*
   - *3.1. TypeID*
   - *3.2. NumIx,*
   - *3.3. CoT*
   - *3.4. OA*
   - *3.5. Address*
   - *3.6. (IOA, DCO)*

En la Figura 3-8 se muestra un pequeño ejemplo de cada uno de los tipos de formatos para que se vea como se debe configurar:

```
('START','auto','uf',STARTDT_ACT) # U-format
('START','auto','if',[45,1,Cause_Act,4,3,(45000,SCO_Off_Np_Se)]) # I-format
('START','auto','sf') } # S-format
```

Figura 3-8: Ejemplo de creación de paquetes de IEC 60870-5-104

Finalmente se implementa un cuerpo principal (*main)* como el de la Figura 3-9. Este *main* consiste en:

- Una primera función que establece la conexión de la siguiente forma: (`iec104_tcp_client(server_ip)`).
- Un bucle que recorre la lista de paquetes que se han configurado previamente y con la función "`client.sendOne(p)`" se transmiten los paquetes.
- Finalmente, se cierra la conexión con la función: `client._socket.close()`.

```
from iec104client import *
from iec104_tcp_packets import *

slave_ip = '192.168.32.129'
client = iec104_tcp_client(server_ip)
client.connect()
for p in packet_list:
    print client.sendOne(p)

client._socket.close()
```

Figura 3-9: *Main* del entorno creado por el alumno

3.3 SCAPY

Scapy es un programa interactivo de manipulación de paquetes, esta herramienta es verdaderamente potente ya que, es capaz de falsificar o decodificar paquetes de un amplio abanico de protocolos. Por esta razón, se utiliza esta librería en la implementación del detector.

Para instalar y poder utilizar la librería se debe ejecutar el siguiente comando en la terminal :

```
pip install scapy
```

En este trabajo se han utilizado distintas funciones de Scapy, las cuales se explican a continuación: (salen muchos avisos, lo único que hago es cambiarte el inicio a cursiva)

- *sniff(count=0, iface=interfaz, prn= función, store=0)*: con esta función se analiza la red en tiempo real.
- *rdcap(traza)*: lee una traza, la almacena en memoria y la guarda en una variable que se va recorriendo para analizar cada paquete uno a uno.
- *pkt.haslayer(protocolo):* esta función se utiliza para averiguar si el paquete analizado contiene dicha capa.
- *pkt[IP].src*: sacar la dirección IP origen.
- *pkt[IP].dst*: : obtener la dirección IP destino.
- *pkt[TCP].sport*: averiguar el puerto de origen.
- *pkt[TCP].dport*: hallar el puerto de destino.
- *pkt[TCP].dataofs*: esta función devuelve el valor de los datos de TCP que al multiplicarlo por 4 se obtiene el valor de la longitud de la cabecera TCP.
- *pkt.show()*: muestra por pantalla el paquete con todos los campos.
- *pkt.START*: con esta función obtenemos el valor del campo de *START* del protocolo IEC 60870-5-104.
- *pkt.ApduLen:* al igual que la función anterior, esta función devuelve el valor de la longitud de la cabecera del protocolo a analizar, pero hay que tener en cuenta que hay que sumarle 2 Bytes más que son lo que ocupan los campos de *START* y *ApduLen*.
- *pkt.Tx*: devuelve el valor del campo *Tx* de la cabecera IEC 60870-5-104.
- *pkt.Rx*: retorna el campo *Rx* de la cabecera IEC 60870-5-104.
- *pkt.TypeID*: se averigua el valor del *TypeID*.
- *pkt.Cause:* función que devuelve el valor del campo *Cause of Transmission*.

3.4 Arquitectura desarrollada

En esta sección, se establecen las pautas que se han seguido a lo largo del proyecto para la implementación del detector, así como la realización de cada parte del código y su funcionamiento.

3.4.1 Programación del detector

Se estableció que el lenguaje de programación que se utilizaría en el proyecto sería Python, ya que es un lenguaje muy intuitivo y de fácil manejo.

Desde el inicio, hay que dejar claro que durante el desarrollo del IDS se ha utilizado un diseño por filtros, de tal forma que si alguno de los campos del protocolo de IEC 60870-5-104 no cumple una de las condiciones establecidas, se considera un ataque, no existiendo término medio.

En primer lugar, se comprueba si el paquete se transporta por TCP/IP, sino cumple esta condición se imprime por pantalla un *WARNING*, mostrando dicho paquete con la función *pkt.show()* para realizar una inspección visual.

El segundo paso, comprueba que alguno de los puertos, ya sea de origen o destino, sea el genérico del protocolo IEC 60870-5-104 y si ninguno de los puertos coincide con el valor 2404, se notifica mediante un *WARNING*, siguiendo las mismas pautas que en el primer paso.

Finalizadas las dos primeras normas, se trata ahora de analizar los distintos campos del protocolo. Los paquetes serán clasificados por formatos. Una vez identificados, se comprueba el campo *ApduLen* en cada uno de los formatos. En los paquetes con formato de supervisión numerada y no numerada, se establece que el campo *ApduLen* sea exactamente 4, de no cumplir esta condición se notifica la anomalía. En cambio, para los paquetes de transferencia de datos dicho campo adquiere un valor que varía desde [4,253].

Tras haber comprobado la longitud de la cabecera APDU, nos centramos sólo en los paquetes de transferencia de datos, en concreto en el campo: *TypeID*. Dicho campo puede tomar valores entre [0,255], el valor <0> no es válido, por lo que si se detecta algún paquete con este *TypeID* se notifica como un ataque. Teniendo en cuenta, que los valores superiores a <128> están reservados para casos excepcionales, por lo que si se identifica algún paquete con alguno de esos valores se activa un WARNING y se notifica por pantalla comprobando si realmente ese usuario tiene derecho a utilizar ese *TypeID* en concreto.

A continuación, se clasifican los paquetes en: pregunta o respuesta. Esta clasificación ha sido comentada previamente en el estado del arte 2.3.1, en donde se detalla que el maestro solo puede enviar tramas (preguntas) para dar órdenes o recopilar información, mientras que los esclavos por su parte solo pueden contestar al maestro (respuestas). Los paquetes provenientes de un maestro pueden tomar los siguientes valores del campo *TypeID*:

{45-51, 58-64, 100-105, 107, 110-113}

En cambio, los paquetes procedentes de un esclavo, los valores del campo *TypeID* deben ser:

{1, 3, 5, 7, 9, 11, 13, 15, 20-21, 30-40, 45-51, 58-64, 70, 100-101, 103-105, 107, 110-113}

Si dicho *TypeID* no coincide con ninguno de los valores mencionados en la dirección correspondiente, se notifica la anomalía mostrando el número del paquete y el campo *TypeID*.

El siguiente campo que se compara es: *Cause of Transmission*, que toma valores entre [0,63]. El valor <0> no está definido, por lo que si se encuentra algún paquete con dicho valor se notifica de esta irregularidad. Además, al igual que en el *TypeID* hay ciertos valores que están reservados para casos especiales, <14-19> y <42-63>, así que se procede con el mismo criterio. Si el campo *CoT* no coinciden con estos valores, <1-13> o <20-41>, se envia un mensaje detallando esta singularidad.

En todo lo anteriormente analizado, hemos tenido en cuenta como detectar una irregularidad a un nivel general, pero a continuación, queremos estudiar de forma específica y detallada, los valores de ciertos campos en concreto.

Dentro de los paquetes de transferencia de datos, el campo *ApduLen* toma valores entre [4,253], pero si el *TypeID* es igual a <45> o <46>, el campo *ApduLen* deja de tener libertad en cuanto al valor a tomar, valiendo exactamente <14>. Si la longitud no cumple la condición, el sistema de detección clasifica dicho paquete como anómalo.

Otra norma implementada en la configuración del IDS es garantizar que para ciertos valores de *TypeID* se debe tener un valor exacto de *CoT*. Siendo los valores del *TypeID*: <45>, <46>, <47>, <48>, <100> y <101>, el campo de *CoT* tomará el valor de <6> en la dirección de control, en cambio, para los mismos valores de *TypeID* pero en la dirección de monitor, el valor de *CoT* puede variar entre <6> o <7>. Si dicha norma no se cumple, se detecta como anomalía indicando el número de paquete, el valor de *CoT* (erróneo) y el supuesto valor que debería de haber tomado en caso de ser correcto.

Además de dejar registro de todas estas anomalías por pantalla, el IDS está programado para guardar dichas irregularidades en un archivo *csv*. También, se guardan en otro archivo las direcciones de los equipos que actúan como maestros y esclavos, con estos archivos se puede comprobar, si alguno de los sistemas que actualmente se encuentra operando en la red no pertenece realmente a ella. Gracias a los archivos *csv* generados por el detector podemos exportar todos los datos obtenidos y analizarlos de forma independiente y así visualizar los datos mediante gráficas, teniendo en cuenta que el proceso de exportación de datos acarrea un tiempo de ejecución mayor.

Para finalizar, cuando el programa termina (ya sea porque el usuario lo decide, pulsando *ctrl+C*, o porque termina de leer una traza) se indica por pantalla:

- el número de paquetes analizados
- el porcentaje de paquetes que utilizan como protocolo de transporte TCP/IP
- los porcentajes de los distintos formatos de IEC 60870-5-104
- una lista de los equipos que han mandado paquetes de transferencia de datos ordenándolos por el campo *TypeID* y un contador que muestra la cantidad de paquetes idénticos que ha transmitido dicho equipo

La forma en que se presentan los resultados no solo simplifica el análisis, sino que también ahorra un tiempo valioso para el analista.

3.4.2 Programación de los ataques

En esta subsección, se explica el procedimiento que se ha seguido a lo largo del proyecto para implementar los ataques. En concreto, se basa en el código explicado en la sección del entorno propio 3.2.4.

En los primeros casos, para cambiar el valor de *ApduLen* se tendrá que ir a la parte del código en la que se da el valor al campo. Como se muestra en la Figura 3-10, sólo habría que modificar el campo sumándole uno para que dicho valor sea erróneo, se subraya la acción para hacerse notar.

```
class i_frame(Packet):
	name = "i_frame"
	fields_desc = [ XByteField("START", 0x68),
			XByteField("ApduLen", None),
			LEShortField("Tx", 0x0000),
			LEShortField("Rx", 0x0000),
			]

	def post_build(self, p, pay):
		if self.ApduLen is None:
			l = len(pay)+4
			p = p[:1] + struct.pack("!B", l) + p[2:]
		return p+pay+1
```

Figura 3-10: Código de la librería de paquetes de transferencia de datos

En cambio, para modificar el valor de *CoT* y poder realizar más ataques, se debe modificar la parte del código de cada tipo de transferencia de datos, como se observa en la Figura 3-11. Dependiendo del valor que se le quiera dar al campo puede significar una cosa u otra, como se ve explicado en el anexo A.B. Se vuelve a subrayar la línea de código exacto que se debería de modificar para su implementación y se tacha el valor que es correcto.

```
class asdu_head(Packet):
	name = "asdu_head"
	fields_desc = [ XByteField("TypeID", 0x45),
			XByteField("SQ", 0x01),
			XByteField("Cause", ~~0x06~~ 0x08),
			XByteField("OA", 0x04),
			LEShortField("Addr", 0x0003)]
```

Figura 3-11: Código del formato transferencia de datos con *TypeID* 45

Con esta configuración se cambian todos los paquetes que se quiere transmitir, pero si se desea modificar un paquete en concreto, habría que cambiar el código de la lista de paquetes que se quiere transmitir, como se muestra en la Figura 3-12. De esta forma, solo se cambiará el valor de *ApduLen* o *CoT*. Además, se podría modificar el valor de *TypeID* para poder mandar paquetes con un valor que no esté implementado. Como se puede ver en

la Figura 3-12, se subrayan los campos que se deben modificar, tachando aquellos valores que debería tomar para una implementación correcta, y cambiándolos por valores erróneos.

```
('START',~~'auto'~~ 16,'if',[~~45~~ 0,1,~~Cause_Act~~ 0x08,4,3,(45000,SCO_Off_Np_Se)])
          ApduLen        TypeID        CoT
```

Figura 3-12: Código de la implementación de la lista de paquetes

También se podría realizar un ataque con una IP que no pertenezca a ninguno de los esclavos o maestros, sino que sea un intruso. Dicho ataque se realizaría cambiando la dirección IP del ordenador con el comando de la Figura 3-13. Y entonces, será el propio detector el que compruebe si dicha IP pertenece a la red y, en caso contrario, notificará este ataque.

```
sudo ifconfig enp0s3 192.168.0.1 netmask 255.255.255.0
```

Figura 3-13: Cambiar la dirección IP

3.5 NetworkMiner

NetworkMiner [25] es una herramienta de análisis de redes de código abierto, disponible para Windows, aunque también existen versiones en Linux. *NetworkMiner* es capaz de analizar archivos PCAP, realizando un estudio que proporciona datos como: direcciones IP, puertos, direcciones MAC de origen y destino. Todo ello en una interfaz de usuario propia como la que se muestra en la Figura 3-14. Dada la gran variedad de protocolos que tiene implementados el *NetworkMiner*, el protocolo IEC 60870-5-104 está entre ellos, por lo que se utiliza como un apoyo para validar y comprobar el detector desarrollado durante este proyecto.

La instalación del *NetworkMiner* es muy sencilla, lo único que hay que hacer es entrar en su página web y descargar el *zip*. Hay dos versiones disponibles, una primera versión que tiene limitaciones y es gratuita, y una segunda opción con todas las configuraciones posibles con un precio de $900. Durante el periodo de desarrollo de este proyecto es suficiente con la versión gratuita, ya que presentaba un abanico de funciones que cumplían con los objetivos del trabajo.

La Figura 3-14 muestra la pantalla de inicio de la herramienta. En el caso que concierne a este TFG, se desea llevar a cabo un análisis de tramas y para ello, se debe acceder a la pestaña *File* y seleccionar la opción de *Open*. Tras ello, aparece una ventana emergente en donde se puede seleccionar el archivo PCAP que se desea analizar y se rellenan los distintos campos existentes, aunque para la comparación que se va a realizar el interés se centra en la pestaña de *Anomalies*, donde se notifican las anomalías detectadas por el programa.

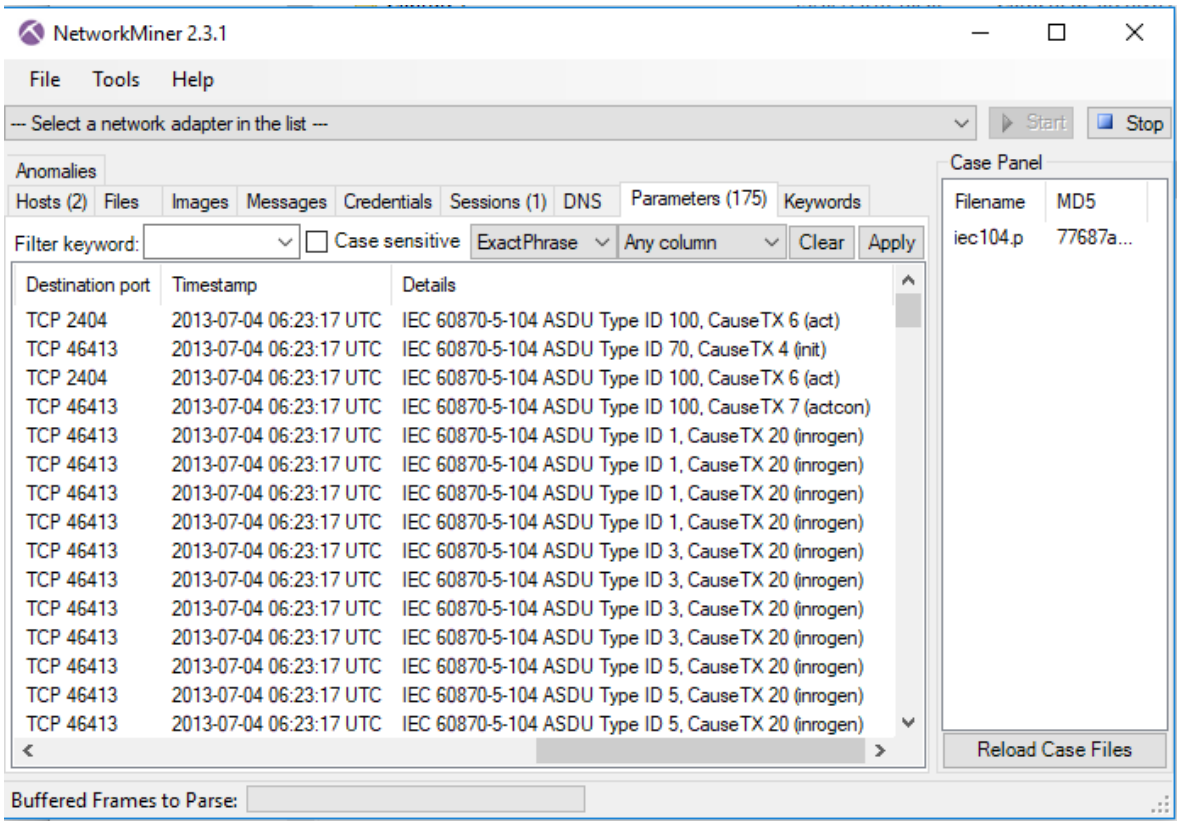


Figura 3-14: NetworkMiner

3.6 Mejoras

Una vez conseguido el correcto funcionamiento de la herramienta desarrollada, se procedió a realizar un estudio sobre posibles mejoras para implementar en el diseño del IDS de forma que el funcionamiento sea más eficiente.

Una de las mejoras que se implementó es cambiar la función "*rdcap(traza)*", por la función "*sniff(offline = traza, prn= función, store=0)*". Con este cambio, lo que se consigue es analizar trazas con gran tamaño de paquetes. El motivo de esta modificación tiene su raíz en la gestión de memoria del sistema, el problema se encuentra en el método que utiliza la función *rdcap* para leer la traza. Esta función, lee la traza entera y la guarda en memoria, haciendo que el análisis sea mucho más eficiente cuando el tamaño de la traza es relativamente pequeño. En cambio, la función *sniff* procede a analizar los paquetes uno a uno y de esta forma se consigue un menor consumo de recursos.

Al finalizar esta mejora, se pueden analizar una amplia gama de trazas, pero en contrapartida el tiempo de ejecución del detector, se ve incrementado. En las primeras pruebas, previas a la implementación de las mejoras, no se llegaba a resultados concluyentes debido a que la función *rdcap* consumía toda la memoria disponible del sistema y provocaba que nunca se alcanzara el final del análisis.

Una vez implementada la primera mejora con la función *Sniff*, actuando sobre las mismas trazas, con una duración aproximada de 30 minutos, el detector tardaba alrededor de 17 horas en conseguir analizar la traza en su totalidad. Esto suponía una clara desventaja, por lo que utilizamos distintos métodos de ejecución, llegando a la conclusión de que la manera óptima de ejecutar nuestro IDS es con *Pypy [24]*, en vez de con *Python*.

Gracias a este método se consiguió disminuir el tiempo de ejecución a 1 hora y 30 minutos, 11 veces más rápido, siendo todavía este tiempo elevado en comparación con la duración de la traza. Como trabajo futuro se puede llegar a conseguir una reducción de los tiempos de ejecución.

Para poder instalar Pypy, es necesario entrar en su página web https://pypy.org/ descargar un archivo binario comprimido y seguir las indicaciones de su web. Una vez descargado Pypy en el equipo, se procedió a instalar el módulo de Scapy, utilizando los siguientes comandos:

1. `Virtualenv --python=/usr/bin/pypy entorno_pypy`
   1.1. Se crea un entorno virtual, en el cual cada vez que se ejecute con Python, se estará ejecutando con Pypy.

2. `Source entorno_pypy/bin/actívate`
   2.1. Se activa dicho entorno.

3. `Pip install scapy`
   `3.1.` Se instala el módulo de Scapy en el nuevo entorno virtual.

3.7 Conclusión

Mediante las normas detalladas en este capítulo, se creó un detector de anomalías sobre el protocolo IEC 60870-5-104, objetivo de este TFG. Al concluir este apartado, se da por finalizado el desarrollo de un IDS y la explicación del entorno donde se trabaja.

En el siguiente capítulo se prueba la efectividad del disector implementado, y se compara con *NetworkMiner*, para ver si dicho disector está bien programado.

4. Validación

4.1 Introducción

En este capítulo se analiza el desarrollo del detector en distintos escenarios virtuales, con el fin de medir su eficacia. Adicionalmente, a través de la Cátedra UAM-Naudit, se facilitó el acceso a una traza, para una validación en un entorno real.

Además, este capítulo es fundamental para extraer conclusiones del proyecto y pensar en mejoras que se puedan realizar en el futuro.

4.2 Ataques realizados

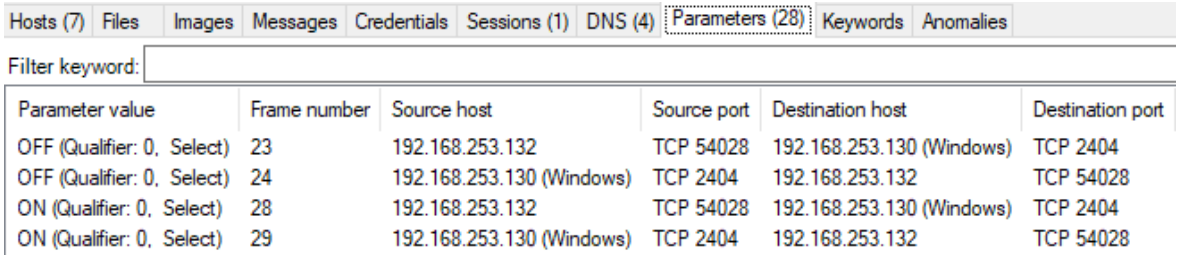
A continuación, se muestran las salidas por terminal de varios entornos en los cuales se han ido modificando parámetros, con el objetivo de estudiar lo que el IDS ha detectado. De esta forma, se verifica que lo que el detector ha conseguido hallar es correcto comparándolo con *NetworkMiner*. En esta parte ha sido necesario crear unas trazas propias con el formato que se explica en el capítulo anterior, 3.2.4.

El primer caso a analizar, es una traza realizada con los programas comentados en el anterior capítulo, 3.2.2 y 3.2.3, con certeza de que es una conexión segura.
Como se puede observar en la Figura 4-1, se muestra una captura de pantalla de lo resultados obtenidos por el detector. En la primera línea detalla el tiempo de ejecución seguido del total de número de paquetes que ha capturado y analizad, acompañado de una breve lista con las direcciones IP del ordenador, tipo de paquete de transferencia de datos enviados y veces se ha repetido dicha combinación. Otro valor que muestra el detector es el porcentaje de paquetes de otros formatos enviados. Y para terminar, en la última línea muestra las anomalías detectadas, en este caso, 0. Al saber que dicha traza se realiza con los programas que establecen una conexión segura del protocolo IEC 60870-5-104, se puede acreditar, que en este caso, el detector ha funcionado según lo establecido, no obstante, dicha traza se compara con el programa de *NetworkMiner* y obtenemos el mismo resultado (Figura 4-2: NetworkMiner primer caso).

```
#### Final results ####
Time elapsed: 0.0676 seconds.
Packets sniffed: 67
---- IEC 60870-5-104 packets Type I: 28  Percent: 90.3225806452 %
             IP           Tipos  Contador     Porcentaje
             192.168.253.132    45     6            21.4285714286 %
             192.168.253.130    45     6            21.4285714286 %
             192.168.253.132    46     7            25.0 %
             192.168.253.130    46     7            25.0 %
             192.168.253.132    47     1            3.57142857143 %
             192.168.253.130    47     1            3.57142857143 %
---- IEC 60870-5-104 packets Type S: 1  Percent: 3.22580645161 %
---- IEC 60870-5-104 packets Type U: 2  Percent: 6.45161290323 %
             STARTDT act = 1
             STARTDT con = 1
             STOPDT act = 0
```

```
            STOPDT con = 0
            TESTFR act = 0
            TESTFR con = 0
---- TCP packets: 51  Percent: 76.1194029851 %
Possible attacks =  0
```

Figura 4-1: Primer caso



Hosts (7) | Files | Images | Messages | Credentials | Sessions (1) | DNS (4) | Parameters (28) | Keywords | Anomalies

Filter keyword:

Parameter value	Frame number	Source host	Source port	Destination host	Destination port
OFF (Qualifier: 0, Select)	23	192.168.253.132	TCP 54028	192.168.253.130 (Windows)	TCP 2404
OFF (Qualifier: 0, Select)	24	192.168.253.130 (Windows)	TCP 2404	192.168.253.132	TCP 54028
ON (Qualifier: 0, Select)	28	192.168.253.132	TCP 54028	192.168.253.130 (Windows)	TCP 2404
ON (Qualifier: 0, Select)	29	192.168.253.130 (Windows)	TCP 2404	192.168.253.132	TCP 54028

Figura 4-2: NetworkMiner primer caso

En el siguiente caso, se modifica el valor de la causa de transmisión comprobando si nuestro detector consigue identificar la anomalía.

Como podemos observar en la Figura 4-3, nuestro detector ha sido bastante rápido en analizar este nuevo caso y los datos que aparecen son muy parecidos a los anteriores, salvo por la última línea en la que el valor cambia y ahora las anomalías detectadas son 18. Al principio de la imagen, el detector imprime la razón por la cual ha clasificado dicho paquete como anómalo.

```
ATTACK wrong CoT, it must be 6 instead of  8 in packet number 29
ATTACK wrong CoT, it must be 7 or 10 instead of  9 in packet number 30
ATTACK wrong CoT, it must be 6 instead of  8 in packet number 32
ATTACK wrong CoT, it must be 7 or 10 instead of  9 in packet number 33
ATTACK wrong CoT, it must be 6 instead of  8 in packet number 35
ATTACK wrong CoT, it must be 7 or 10 instead of  9 in packet number 36
ATTACK wrong CoT, it must be 6 instead of  8 in packet number 38
ATTACK wrong CoT, it must be 7 or 10 instead of  9 in packet number 39
ATTACK wrong CoT, it must be 6 instead of  8 in packet number 41
ATTACK wrong CoT, it must be 6 instead of  8 in packet number 43
ATTACK wrong CoT, it must be 6 instead of  8 in packet number 45
ATTACK wrong CoT, it must be 6 instead of  8 in packet number 47
ATTACK wrong CoT, it must be 6 instead of  8 in packet number 49
ATTACK wrong CoT, it must be 6 instead of  8 in packet number 51
ATTACK wrong CoT, it must be 6 instead of  8 in packet number 53
ATTACK wrong CoT, it must be 6 instead of  8 in packet number 55
ATTACK wrong CoT, it must be 6 instead of  8 in packet number 57
ATTACK wrong CoT, it must be 6 instead of  8 in packet number 59


#### Final results ####
Time elapsed: 0.0829 seconds.
Packets sniffed: 63
---- IEC 60870-5-104 packets Type I: 30  Percent: 90.9090909091 %
            IP           Tipos  Contador     Porcentaje
            192.168.253.128     45     6            20.0 %
```

```
            192.168.253.129    45    6           20.0 %
            192.168.253.128    46    7           23.3333333333 %
            192.168.253.129    46    4           13.3333333333 %
            192.168.253.128    47    7           23.3333333333 %
---- IEC 60870-5-104 packets Type S: 1  Percent: 3.0303030303 %
---- IEC 60870-5-104 packets Type U: 2  Percent: 6.06060606061 %
            STARTDT act = 1
            STARTDT con = 1
            STOPDT act = 0
            STOPDT con = 0
            TESTFR act = 0
            TESTFR con = 0
---- TCP packets: 59  Percent: 93.6507936508 %
Possible attacks =  18
```

Figura 4-3: Segundo caso

Como podemos ver en la Figura 4-3, el detector revela que en varios paquetes el valor de *CoT* es erróneo mostrando por pantalla el valor que debería tomar y a continuación, el valor erróneo. En cambio, la misma traza analizada por el programa *NetworkMiner*, no consigue detectar la anomalía como se muestra en la Figura 4-4.

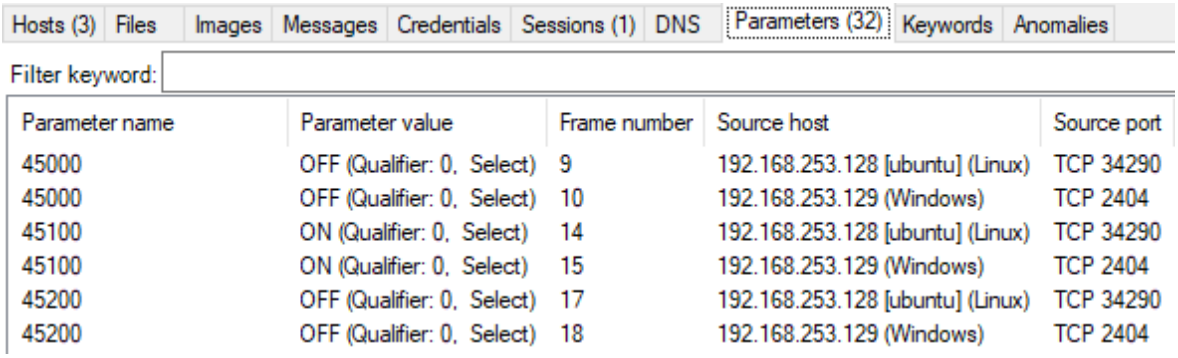


Figura 4-4: *NetworkMiner* segundo caso

A continuación, modificamos el valor de longitud de la cabecera APDU, implementando un código, el cual envíe un paquete de transferencia de datos del tipo 45 con una longitud de 16. Según la explicación en el capítulo de desarrollo, el detector debe de identificar la anomalía.

Como se observa en la Figura 4-5, el detector ha sido capaz de identificar las irregularidades que se querían demostrar. Mostrando por pantalla que, para un cierto número de paquetes, el valor de la longitud debería ser 14, pero en realidad toma el valor 16, detectando la anomalía.

```
ATTACK wrong ApduLen, it must be 14 instead of  16 in packet number 9
ATTACK wrong ApduLen, it must be 14 instead of  16 in packet number 13
ATTACK wrong ApduLen, it must be 14 instead of  16 in packet number 15
ATTACK wrong ApduLen, it must be 14 instead of  16 in packet number 17
ATTACK wrong ApduLen, it must be 14 instead of  16 in packet number 19
ATTACK wrong ApduLen, it must be 14 instead of  16 in packet number 21
ATTACK wrong ApduLen, it must be 14 instead of  16 in packet number 25
```

```
ATTACK wrong CoT, it must be 6 instead of  8 in packet number 25

#### Final results ####
Time elapsed: 0.0098 seconds.
Packets sniffed: 26
---- IEC 60870-5-104 packets Type I: 7  Percent: 70.0 %
             IP           Tipos  Contador     Porcentaje
             192.168.253.128     45     6            85.7142857143 %
             192.168.253.128     46     1            14.2857142857 %
---- IEC 60870-5-104 packets Type S: 1  Percent: 10.0 %
---- IEC 60870-5-104 packets Type U: 2  Percent: 20.0 %
             STARTDT act = 1
             STARTDT con = 1
             STOPDT act = 0
             STOPDT con = 0
             TESTFR act = 0
             TESTFR con = 0
---- TCP packets: 24  Percent: 92.3076923077 %
Possible attacks =  8
```

Figura 4-5: Tercer caso

En cambio, el análisis realizado con el *NetworkMiner* se muestra en la Figura 4-6, en comparación con el IDS, este no consigue detectar ninguna anomalía, bien porque desde que recibe el paquete y ve que el receptor lo ha rechazado, no lo identifica como una anomalía, o porque dicha norma no está programada.

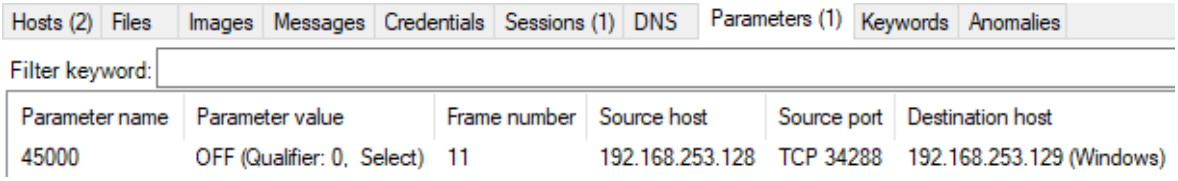


Figura 4-6: *NetworkMiner* tercer caso

Hasta ahora la creación de dichos entornos, se ha hecho de tal forma de que todos los paquetes eran iguales. En el siguiente caso se realiza sólo un cambio para un paquete específico, viendo si el detector es capaz de identificar en que paquete se encuentra dicha anomalía, comparándolo con la traza para verificarlo.

Como se puede ver en la Figura 4-7, el detector ha sido bastante rápido en analizar esta trama, y ha conseguido hallar un error en un paquete.

```
ATTACK wrong ApduLen, it must be 14 instead of  15 in packet number 26
ATTACK wrong CoT, it must be 6 instead of  8 in packet number 26
ATTACK wrong CoT, it must be 6 instead of  8 in packet number 28
ATTACK wrong CoT, it must be 6 instead of  8 in packet number 30
ATTACK wrong CoT, it must be 6 instead of  8 in packet number 34

#### Final results ####
Time elapsed: 0.0342 seconds.
Packets sniffed: 38
---- IEC 60870-5-104 packets Type I: 15  Percent: 83.3333333333 %
             IP           Tipos  Contador     Porcentaje
             192.168.253.129     45     6            40.0 %
```

```
            192.168.253.128     45     5            33.3333333333 %
            192.168.253.129     46     4            26.6666666667 %
---- IEC 60870-5-104 packets Type S: 1  Percent: 5.55555555556 %
---- IEC 60870-5-104 packets Type U: 2  Percent: 11.1111111111 %
            STARTDT act = 1
            STARTDT con = 1
            STOPDT act = 0
            STOPDT con = 0
            TESTFR act = 0
            TESTFR con = 0
---- TCP packets: 35  Percent: 92.1052631579 %
Possible attacks =  5
```

Figura 4-7: Cuarto caso

Debemos de comprobar si el detector ha hecho un análisis correcto comparándolo con el *NetworkMiner*. La Figura 4-8 muestra que el programa sigue sin detectar ninguna anomalía, por lo que utilizaremos el programa de Wireshark para acreditar que dicha anomalía sí que es detectada.

Hosts (3) | Files | Images | Messages | Credentials | Sessions (1) | DNS | Parameters (11) | Keywords | Anomalies

Filter keyword:

Parameter name	Parameter value	Frame number	Source host	Source port	Destination host
45000	OFF (Qualifier: 0, Select)	7	192.168.253.129	TCP 52532	192.168.253.128 (Windows)
45000	OFF (Qualifier: 0, Select)	8	192.168.253.128 (Windows)	TCP 2404	192.168.253.129
45100	ON (Qualifier: 0, Select)	12	192.168.253.129	TCP 52532	192.168.253.128 (Windows)
45100	ON (Qualifier: 0, Select)	13	192.168.253.128 (Windows)	TCP 2404	192.168.253.129
45200	OFF (Qualifier: 0, Select)	15	192.168.253.129	TCP 52532	192.168.253.128 (Windows)
45200	OFF (Qualifier: 0, Select)	16	192.168.253.128 (Windows)	TCP 2404	192.168.253.129
45300	ON (Qualifier: 0, Select)	18	192.168.253.129	TCP 52532	192.168.253.128 (Windows)
45300	ON (Qualifier: 0, Select)	19	192.168.253.128 (Windows)	TCP 2404	192.168.253.129
45400	OFF (Qualifier: 0, Select)	21	192.168.253.129	TCP 52532	192.168.253.128 (Windows)
45400	OFF (Qualifier: 0, Select)	22	192.168.253.128 (Windows)	TCP 2404	192.168.253.129
45500	ON (Qualifier: 0, Select)	26	192.168.253.129	TCP 52532	192.168.253.128 (Windows)

Figura 4-8: *NetworkMiner* cuarto caso

En cambio, cuando analizamos la trama con el *Wireshark* se puede observar que en la Figura 4-9, sí que detecta que hay una malformación en el paquete debido a un valor de *ApduLen* inválido, certificando que la configuración del detector es correcta.

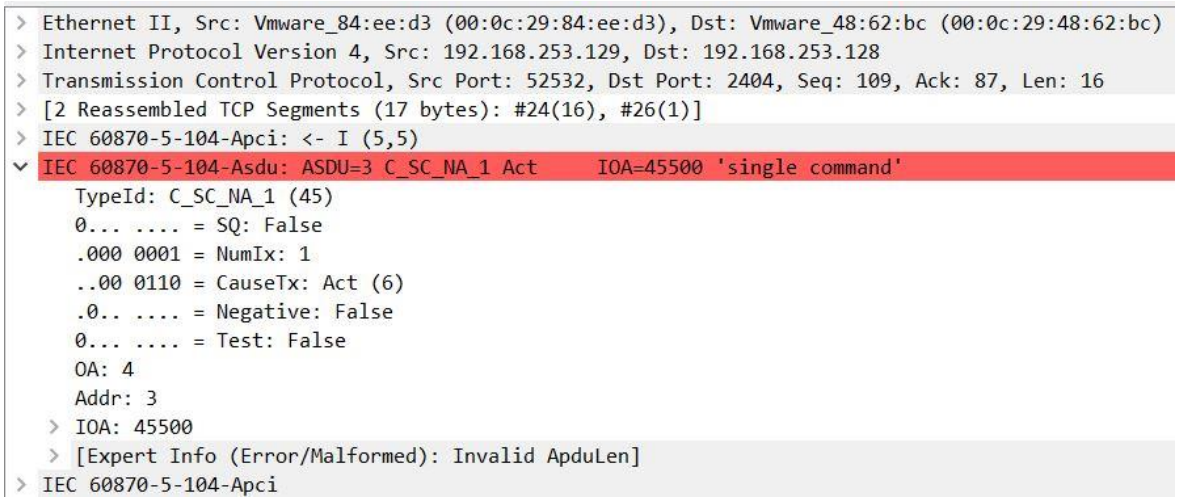


Figura 4-9: Wireshark cuarto caso

En la Figura 4-10, se muestra lo que se ve por pantalla, si se identifica un paquete que no se transporta por TCP/IP, no se puede identificar como un ataque ni como una anomalía por lo que el detector imprime por pantalla dicho paquete y el usuario tiene que valorar si realmente el paquete es maligno. Por lo que siempre que se den estas situaciones, se imprimen todos los valores de cada uno de los protocolos que contiene dicho paquete por pantalla.

```
WARNING NO TCP
###[ Ethernet ]###
  dst       = 00:50:56:ed:8a:00
  src       = 00:0c:29:48:62:bc
  type      = 0x800
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 96
     id        = 922
     flags     =
     frag      = 0
     ttl       = 128
     proto     = udp
     chksum    = 0xbb1e
     src       = 192.168.253.128
     dst       = 192.168.253.2
     \options   \
###[ UDP ]###
        sport     = netbios_ns
        dport     = netbios_ns
        len       = 76
        chksum    = 0xb6ea
###[ NBNS query request ]###
           NAME_TRN_ID= 61649
           FLAGS     = 16384
           QDCOUNT   = 1
           ANCOUNT   = 0
```

```
            NSCOUNT   = 0
            ARCOUNT   = 1
            QUESTION_NAME= '\x01\x02__MSBROWSE__\x02'
            SUFFIX    = 16706
            NULL      = 0
            QUESTION_TYPE= NB
            QUESTION_CLASS= INTERNET
###[ Raw ]###
               load      = '\xc0\x0c\x00
\x00\x01\x00\x04\x93\xe0\x00\x06\xe0\x00\xc0\xa8\xfd\x80'
```

Figura 4-10: Warning

En este último caso se analizó una traza en la cual había una gran variedad de paquetes de transferencia de datos con distintos tipos, y como se puede observar al final de la Figura 4-11, dicha traza, el detector la clasificó como una conexión segura. Esta traza se obtuvo de Internet, de la página web [15].

```
#### Final results ####
Time elapsed: 0.1116 seconds.
Packets sniffed: 105
---- IEC 60870-5-104 packets Type I: 104  Percent: 81.25 %
             IP                 Tipos  Contador     Porcentaje
             10.20.100.108      5      14           13.4615384615 %
             10.20.100.108      30     9            8.65384615385 %
             10.20.100.108      34     8            7.69230769231 %
             10.20.102.1        45     2            1.92307692308 %
             10.20.100.108      45     7            6.73076923077 %
             10.20.102.1        46     2            1.92307692308 %
             10.20.100.108      46     7            6.73076923077 %
             10.20.102.1        47     2            1.92307692308 %
             10.20.100.108      47     8            7.69230769231 %
             10.20.102.1        48     2            1.92307692308 %
             10.20.100.108      48     7            6.73076923077 %
             10.20.102.1        49     2            1.92307692308 %
             10.20.100.108      49     7            6.73076923077 %
             10.20.102.1        50     2            1.92307692308 %
             10.20.100.108      50     7            6.73076923077 %
             10.20.102.1        51     2            1.92307692308 %
             10.20.100.108      51     8            7.69230769231 %
             10.20.100.108      70     1            0.961538461538 %
             10.20.102.1        100    2            1.92307692308 %
             10.20.100.108      100    5            4.80769230769 %
---- IEC 60870-5-104 packets Type S: 14  Percent: 10.9375 %
---- IEC 60870-5-104 packets Type U: 10  Percent: 7.8125 %
             STARTDT act = 1
             STARTDT con = 1
             STOPDT act = 0
             STOPDT con = 0
             TESTFR act = 4
             TESTFR con = 4
---- TCP packets: 105  Percent: 100.0 %
Possible attacks =  0
```

Figura 4-11: Traza real de internet

Analizando si dicha traza era segura como había identificado nuestro detector, se compara con el programa *NetworkMiner*, como se había hecho hasta ahora, Figura 4-12. También se certificó que en *Wireshark* no detecta ningún campo inválido. Habiendo comprobado con esos dos programas que los resultados obtenidos eran los mismos en los tres casos, la efectividad del analizador quedó más que comprobada.

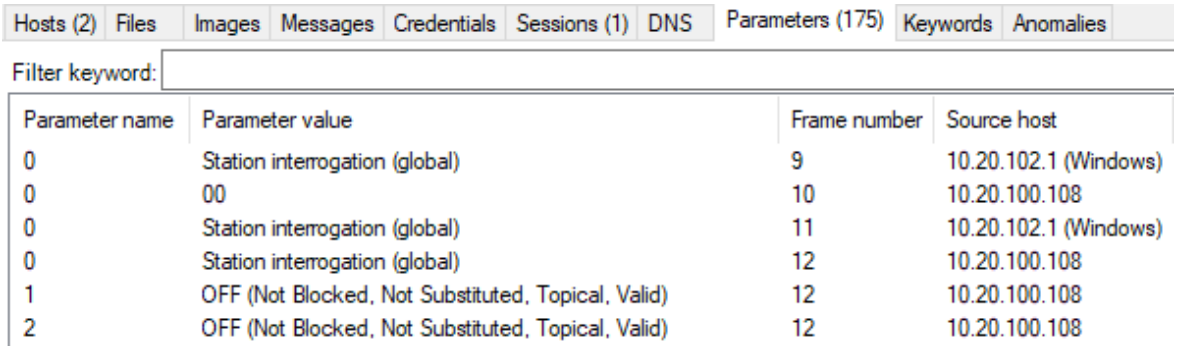


Figura 4-12: NetworkMiner captura de internet

En la siguiente sección se explicará una de las validaciones más importantes, ya que se hará sobre un entrono real. Hasta ahora, era muy difícil ver como era la jerarquía de este protocolo, pero con este caso se demostrará lo estudiado en el estado del arte.

4.3 Caso real

En este capítulo se va a proceder a explicar con detenimiento un caso real, para el cual la Cátedra UAM-Naudit, ayudó a validar dicho trabajo. Debido a temas de confidencialidad hay varios campos que se omitirán no pudiéndose mostrar cómo es el caso de las direcciones IP de los equipos.

Gracias a esta traza, se realizó la mejora de cambiar la función comentada en el capítulo anterior, en la sección de mejoras 3.6. El tamaño de la traza era tan grande que el detector no terminaba de ejecutar.

Una vez realizado el cambio, se obtuvieron los ficheros correspondientes, los cuales nos muestran como podría ser la arquitectura de una empresa real, observando que un maestro puede tener más de 100 esclavos a la vez. Gracias a estos datos, se verificó que lo que se veía en el capítulo del estado del arte era cierto, y que no afloró en la sección anterior debido a su complejidad.

A continuación, se muestra por pantalla, lo que el detector imprimió:

```
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 69789
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 148466
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 226605
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 305807
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 385331
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 464555
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 544364
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 622953
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 702464
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 782132
```

```
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 861767
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 941349
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 1020917
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 1100351
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 1179565
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 1259495
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 1339890
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 1420219
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 1501182
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 1582068
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 1660605
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 1739392
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 1819466
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 1820462
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 1899541
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 1980022
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 2061454
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 2141319
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 2221550
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 2301598
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 2380694
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 2460766
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 2541680
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 2621174
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 2700939
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 2781396
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 2861879
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 2942137
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 3023079
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 3102126
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 3181333
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 3262484
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 3342119
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 3421036
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 3500836
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 3580407
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 3659278
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 3739821
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 3820081
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 3899420
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 3980694
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 4060394
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 4140649
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 4221859
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 4302021
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 4381306
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 4461670
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 4541002
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 4621183
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 4701853
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 4780420
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 4860689
ATTACK wrong CoT, it must be 7 or 10 instead of  46 in packet number 4942817

#### Final results ####
Time elapsed: 50049.5514 seconds.
Packets sniffed: 5000000
---- IEC 60870-5-104 packets: 2602163  Percent: 52.04326 %
---- IEC 60870-5-104 packets Type I: 736919  Percent: 28.3194788336 %
```

```
--- IEC 60870-5-104 packets Type S: 337098  Percent: 12.9545305194 %
---- IEC 60870-5-104 packets Type U: 1528142  Percent: 58.7258369287 %
             STARTDT act = 1981
             STARTDT con = 668
             STOPDT act = 0
             STOPDT con = 0
             TESTFR act = 775032
             TESTFR con = 750461
---- TCP packets: 5000000  Percent: 100.0 %
Possible attacks =  67
```

Figura 4-13: Caso real

Como se puede analizar en la Figura 4-13, vemos que el detector identificó 67 casos en los cuales encontró una anomalía en la traza. Si miramos más arriba, la razón por el cual el detector identifico esos paquetes como anómalos, es debido a que el valor de la causa de transmisión no era el adecuado, y siempre tomaba el valor <46>.

Buscando en la tabla de valores de la causa de transmisión, A.B, se observó que el valor <46> correspondía a que no conocía la dirección ASDU. Por lo que se llegó a la siguiente conclusión:

Analizando la traza de 5.000.000 de paquetes del protocolo IEC 60870-5-104, se puede observar que un maestro, envía de forma continuada durante prácticamente la duración de toda la traza paquetes del tipo I-format a un cierto esclavo, con un TypeID de valor 100 (Comando de interrogación C_IC_NA_1) y con una dirección ASDU desconocida para el esclavo, por lo que justifica que al responder el valor de CoT sea 46.

Además, también es adecuado mencionar el abundante número de paquetes U-format que abarcan el 59% del total del protocolo IEC 60870-5-104, y es causado en el 99,8% de las veces por el tráfico de paquetes TESTFR, que son tramas para testear que la conexión sigue activa entre maestro y esclavo (el 0.2% restante es de tramas de establecimiento de sesión).

4.4 Conclusión

Los resultados de las distintas pruebas realizadas sobre el IDS durante la fase de validación permiten afirmar que la implementación realizada en el capítulo 3, cumple con lo establecido. Obteniendo unos resultados de forma que al analista le sea de utilidad y pueda tomar decisiones con más datos que sostengan resolución.

Este capítulo, coge gran importancia gracias a la sección 4.3, en el cual se ve un caso real y se explica lo que el detector ha sido capaz de analizar, pudiendo observar lo que se explica durante el estado del arte aplicado a un entorno real.

5. Conclusiones y trabajo futuro

5.1 Conclusión

El objetivo de este Trabajo Fin de Grado era el desarrollo de un sistema de detección de tráfico anómalo en redes SCADA basadas en el protocolo IEC 60870-5-104. Dicho objetivo ha quedado demostrado con el capítulo de validaciones. Este proyecto está subido en la página de *Github,* https://github.com/davideto6/IEC-60870-5-104 ,para que cualquier usuario que lo desea pueda utilizarlo.

Las ideas principales que aporta este trabajo al ámbito académico son: una explicación de cómo trabajar con la librería Scapy, un manual de como instalar máquinas virtuales, ya sea para este uso en particular u otro uso cualquiera, y sobre todo un acercamiento a las redes SCADA y su funcionamiento en especial para el protocolo IEC 60870-5-104. Además, se explica que otros métodos de ejecución como es el caso de Pypy para

Con este trabajo me gustaría compartir y hacer ver a todos mis compañeros que en los trabajos fin de grado se puede tener la suerte como es en mi caso, de poder validar tu trabajo con algo real, y tal hecho crea una motivación que no se puede explicar.

Se decidió escoger un trabajo de fin de grado sobre telemática debido al interés sobre las redes que ha ido creciendo a medida que se iban realizando las asignaturas de dicha rama. En cuanto al tema del trabajo, se hizo una tutoría para ver que opciones había disponibles y nos decantamos por ciberseguridad. Al principio, era un tema en el que no me manejaba mucho, pero a medida que he ido investigando y aprendiendo más del ámbito me empecé a sentir más cómodo, por eso motivo a los alumnos a que no tengan miedo a lanzarse a realizar un proyecto en el cual al principio no se domine muy bien, no hay nada que con trabajo y esfuerzo no se pueda realizar.

En cuanto a mis conocimientos adquiridos a lo largo de este trabajo, han sido muy instructivos, gracias a mi tutor y a los profesores que me han ayudado, he aprendido a programar con Python con mucha soltura, y sobre todo numerosos comandos de la terminal que me han ayudado a avanzar más rápido en mi proyecto. Si tuviera que destacar algo, sería el manejo de las máquinas virtuales y la terminal para su configuración.

Para terminar, la realización de este trabajo ha supuesto una satisfacción personal, además de porque dicho trabajo fue probado en un entorno real, por el compromiso y la dedicación de ciertos profesores de la propia universidad que quisieron ayudar en este proyecto, a los cuales doy las gracias.

5.2 Trabajo Futuro

Como trabajo futuro de este TFG se podrían intentar varias cosas. La principal mejora que se debería aportar al proyecto sería poder reducir el tiempo de ejecución hasta conseguir que dicho tiempo se igual o menor al tiempo de la traza. Otro avance sería la creación de un entorno en el cuál se pudiese crear distintas máquinas virtuales y establecer cual

actuaría como maestro y cual como esclavo. Con la creación de ese entorno un poco complejo, se podrían crear conexiones sin limitaciones de paquetes y de configuración de valores.

Además, se podría también comprobar los números de secuencia de TCP para verificar que no existe ninguna anomalía, así como los campos de *Tx* y *Rx* de la cabecera APCI, los cuales tienen la misma función en el protocolo del estudio.

Otro trabajo futuro, podría ser modificar el código del IDS para convertirlo en un IPS pudiendo detectar los ataques y que él sólo los erradique. Este punto se planteó, y no se llevó a cabo, durante la implementación del disector, que cada vez que detectará un ataque, emitiera un paquete en ambas direcciones con un RESET para reiniciar la conexión entre ambos.

Todas estas mejoras podrían ayudar al estudio sobre este protocolo, y en especial a las empresas que lo tienen implementado en sus redes.

Referencias

[1] IEC, " Telecontrol equipment and systems – Part 5-104 Transmission protocols-Network access for IEC 60870-5-101 using standard transport profiles", Junio 2006 https://webstore.iec.ch/preview/info_iec60870-5-104%7Bed2.0%7Den_d.pdf

[2] Carl Gould, "What is SCADA?", Junio 2017 https://inductiveautomation.com/what-is-scada

[3] IPCOMM GmbH "IEC60870-5-104", Noviembre 2017 www.ipcomm.de/protocol/IEC104/en/sheet.html

[4] Adif, "Protocolo de Comunicaciones entre el centro de control y remotas en telemandos de energía de líneas de alta velocidad", abril 2018 http://www.adif.es/es_ES/empresas_servicios/normativa_tecnica/doc/ET033595018.pdf

[5] Cyrille Larrieu, "Sistemas de detección de intrusiones", Enero 2003 https://es.ccm.net/contents/162-sistema-de-deteccion-de-intrusiones-ids#introduccion-a-los-sistemas-de-deteccion-de-intrusiones

[6] Maxim, "What is the definition of "PLC"?", *Noviembre 2017* https://unitronicsplc.com/what-is-plc-programmable-logic-controller/

[7] Sayer Qaisar, "What is SCADA?", Septiembre 2012 https://www.integraxor.com/what-is-scada/

[8] Dharma Teja, "Master Terminal Unit (MTU) in SCADA systems", Junio 2011 http://electricalquestionsguide.blogspot.com/2011/06/master-terminal-units-mtu-in-scada.html

[9] David Kusher, "The real story of Stuxnet", Febrero 2013 https://spectrum.ieee.org/telecom/security/the-real-story-of-stuxnet

[10] Mattias Eriksson, "An example of a Man-in-the-middle Attack Against Server Authenticated SSL-sessions", Mayo 2018 http://www.ethikalhacker.webs.com/ManInTheMiddle_SSL.pdf

[11] Stretch, "Understanding TCP Sequence and Acknowledgment Numbers", Junio 2010 http://packetlife.net/blog/2010/jun/7/understanding-tcp-sequence-acknowledgment-numbers/

[12] Guadalupe Moreno, "Los ciberataques en España", Noviembre 2011 https://es.statista.com/grafico/6876/los-ciberataques-en-espana/

[13] Huawei, "Attack Defense Configuration" , Mayo 2018 http://support.huawei.com/enterprise/docinforeader!loadDocument1.action?contentId=DOC1000019451&partNo=100132

[14] Chris Hoffman, "What is the Man-in-the-middle Attack? Security Jargon explained", Julio 2014 https://www.makeuseof.com/tag/man-middle-attack-security-jargon-explained/

[15] Automayt, "ICS-pcap", Junio 2016 https://github.com/automayt/ICS-pcap/tree/master/IEC%2060870/iec104

[16] Deloitte, "¿Qué es la industria 4.0?", Enero 2018 https://www2.deloitte.com/es/es/pages/manufacturing/articles/que-es-la-industria-4.0.html

[17] Y. Yang, K. McLaughlin, T. Littler, S. Sezer, B. Pranggono, Intrusion "Detection System for IEC 60870-5-104 Based SCADA Networks", año 2013 https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6672100

[18] Faculty of Information Technology Brno University of Technology "Description and analysis of IEC 104 Protocol", *Diciembre 2017* *http://www.fit.vutbr.cz/research/pubs/all.php?file=%2Fpub%2F11570%2FTR-IEC104.pdf&id=11570*
[19] Werner Mayr, "IEC 60870-5-104: Telegram Structure", Junio 2016, http://www.mayor.de/lian98/doc.en/html/u_iec104_struct.htm
[20] Ricardo Olsen, QTester104, Diciembre 2017 https://sourceforge.net/projects/qtester104/
[21] WinPP104 http://winpp104.software.informer.com/17.0/
[22] VMware https://www.vmware.com/es/products.html
[23] Scapy https://scapy.net/
[24] Pypy http://pypy.org/
[25] NetworkMiner http://www.netresec.com/?page=NetworkMiner
[26] Durga Samanth, Rajesh Kalluri, R. K. Senthil, B. S. Bindhumadhva, "SCADA communication protocols: vulnerabilities, attacks and possible mitigations", Abril 2013 https://link.springer.com/article/10.1007/s40012-013-0013-5

Anexos

A Tipos de identificación

TypeID	Utilidad	Nomenclatura
1	Información de punto simple	M_SP_NA_1
2	Información de punto simple con etiqueta de tiempo	M_SP_TA_1
3	Información de punto doble	M_DP_NA_1
4	Información de punto doble con etiqueta de tiempo	M_DP_TA_1
5	Información sobre la posición	M_ST_NA_1
6	Información sobre la posición con etiqueta de tiempo	M_ST_TA_1
7	Cadena de 32 bits	M_BO_NA_1
8	Cadena de 32 bits con etiqueta de tiempo	M_BO_TA_1
9	Valor medido normalizado	M_ME_NA_1
10	Valor medido normalizado con etiqueta de tiempo	M_ME_TA_1
11	Valor medido escalado	M_ME_NB_1
12	Valor medido escalado con etiqueta de tiempo	M_ME_TB_1
13	Valor medido short floating	M_ME_NC_1
14	Valor medido short floating con etiqueta de tiempo	M_ME_TC_1
15	Totales integrados	M_IT_NA_1
16	Totales integrados con etiqueta de tiempo	M_IT_TA_1
17	Evento de equipo de protección con etiqueta de tiempo	M_EP_TA_1
18	Eventos de inicio empaquetados del equipo de protección con etiqueta de tiempo	M_EP_TB_1
19	Información del circuito de salida del equipo de protección con etiqueta de tiempo.	M_EP_TC_1
20	Información empaquetada de un solo punto con detección de cambio de estado	M_PS_NA_1
21	Valor medido normalizado sin descriptor de calidad	M_ME_ND_1
30	Información de punto simple con etiqueta de tiempo CP56Time2a	M_SP_TB_1
31	Información de punto doble con etiqueta de tiempo CP56Time2a	M_DP_TB_1
32	Información sobre la posición con etiqueta de tiempo CP56Time2a	M_ST_TB_1
33	Cadena de 32 bits con etiqueta de tiempo CP56Time2a	M_BO_TB_1
34	Valor medido normalizado con etiqueta de tiempo CP56Time2a	M_ME_TD_1
35	Valor medido escalado con etiqueta de tiempo CP56Time2a	M_ME_TE_1
36	Valor medido con short floating con etiqueta de tiempo CP56Time2a	M_ME_TF_1
37	Total de integrantes con etiqueta de tiempo CP56Time2a	M_IT_TB_1
38	Evento de equipo de protección con etiqueta de tiempo CP56Time2a	
39	Eventos de inicio empaquetados del equipo con etiqueta de tiempo CP56Time2a	M_EP_TE_1
40	Información del circuito de salida del equipo con etiqueta de tiempo CP56Time2a	M_EP_TF_1
45	Comando simple	C_SC_NA_1
46	Comando doble	D_DC_NA_1
47	Comando regulador de posición	C_RC_NA_1

48	Comando setpoint, valor normalizado	C_SE_NA_1
49	Comando setpoint, valor escalado	C_SE_NB_1
50	Comando setpoint, short floating	C_SE_NC_1
51	Cadena de 32 bits	C_BO_NA_1
58	Comando simple con etiqueta de tiempo CP56Time2a	C_SC_TA_1
59	Comando doble con etiqueta de tiempo CP56Time2a	C_DC_TA_1
60	Comando regulador de posición con etiqueta de tiempo CP56Time2a	C_RC_TA_1
61	Comando setpoint, valor normalizado con etiqueta de tiempo CP56Time2a	C_SE_TA_1
62	Comando setpoint, valor escalado con etiqueta de tiempo CP56Time2a	C_SE_TB_1
63	Comando setpoint, short floating con etiqueta de tiempo CP56Time2a	C_SE_TC_1
64	Cadena de 32 bits con etiqueta de tiempo CP56Time2a	C_BO_TA_1
70	Fin de la inicialización	M_EI_NA_1
100	Comando de interrogación	C_IC_NA_1
101	Comando contador de interrogación	C_CI_NA_1
102	Comando de lectura	C_RD_NA_1
103	Comando de sincronización de reloj	C_CS_NA_1
104	Comando de prueba	C_TS_NB_1
105	Comando de reseteo de procesos	C_RP_NC_1
106	Comando de retardo de adquisición	C_CD_NA_1
107	Comando de prueba con etiqueta de tiempo CP56Time2a	C_TS_TA_1
110	Parámetros de valor normalizados	P_ME_NA_1
111	Parámetros de valor escalados	P_ME_NB_1
112	Parámetros de valor short floating	P_ME_NC_1
113	Activación de parámetros	P_AC_NA_1
120	Archivo preparado	F_FR_NA_1
121	Sección preparada	F_SR_NA_1
122	Llamada al directorio, selección del archivo, llamada al archivo, llamada a la sección	F_SC_NA_1
123	Última sección, último segmento	F_LS_NA_1
124	Archivo ACK, sección ACK	F_AF_NA_1
125	Segmento	F_SG_NA_1
126	Directorio	F_DR_TA_1
127	QueryLog – Solicitar fichero de archivo	F_SC_NB_1

B Causa de transmisión

CoT	Utilidad	Nomenclatura
1	Periódico, cíclico	
2	Interrogante de fondo	
3	Espontáneo	
4	Inicialización	Init
5	Interrogación o interrogante	Req
6	Activación	Act
7	Confirmación de la activación	Actcon
8	Desactivación	Deact
9	Confirmación de la desactivación	Deactcon
10	Final activación	Actterm
11	Retroalimentación, causada por un comando distante	
12	Retroalimentación, causada por un comando local	
13	Transmisión de datos	
20	Interrogado por interrogación general	Inrogen
21	Interrogado por interrogación del grupo 1	
22	Interrogado por interrogación del grupo 2	
23	Interrogado por interrogación del grupo 3	
24	Interrogado por interrogación del grupo 4	
25	Interrogado por interrogación del grupo 5	
26	Interrogado por interrogación del grupo 6	
27	Interrogado por interrogación del grupo 7	
28	Interrogado por interrogación del grupo 8	
29	Interrogado por interrogación del grupo 9	
30	Interrogado por interrogación del grupo 10	
31	Interrogado por interrogación del grupo 11	
32	Interrogado por interrogación del grupo 12	
33	Interrogado por interrogación del grupo 13	
34	Interrogado por interrogación del grupo 14	
35	Interrogado por interrogación del grupo 15	
36	Interrogado por interrogación del grupo 16	
37	Interrogado por un contador de interrogación general	
38	Interrogado por un contador de interrogación por el grupo 1	
39	Interrogado por un contador de interrogación por el grupo 2	
40	Interrogado por un contador de interrogación por el grupo 3	
41	Interrogado por un contador de interrogación por el grupo 4	
44	Tipo de identificación desconocido	
45	Causa desconocida	
46	Dirección ASDU desconocida	
47	Dirección de la información del objeto desconocida	